UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Reconhecimento de Caracteres Alfanuméricos de Placas em Imagens de Veículos

por
TATIANE JESUS DE CAMPOS

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Professor Dr. Sergio Bampi
Orientador

Professor Dr. Altamiro Amadeu Suzim
Co- orientador

Porto Alegre, 23 de agosto de 2001.

CIP – CATALOGAÇÃO NA PUBICAÇÃO

Campos, Tatiane Jesus de
Reconhecimento de caracteres alfanuméricos de placas em imagens de veículos / por Tatiane Jesus de Campos. – Porto Alegre: PPGC da UFRGS, 2001.
122p.
Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação, Porto Alegre, BR – RS, 2001. Orientador: Bampi, Sergio; Co-orientador: Suzim, Altamiro Amadeu.
1. identificação automática de veículos. 2. processamento de imagem. 3. redes neurais. I. Bampi, Sergio. II. Suzim, Altamiro Amadeu. III. Título

Dedico esta dissertação à minha família pela compreensão e dedicação e ao Mario pelo constante incentivo.

Agradecimentos

Agradeço inicialmente a todos que contribuíram no desenvolvimento de minha formação. À minha família, pela cultura herdada e pelos valores que me infundiram o caráter, e às instituições de ensino que me transmitiram desde cedo o saber e o gosto pela ciência.

Ao meu orientador Sergio Bampi, pela sua orientação e confiança depositada e também pela visão otimista, incentivo e apoio nas mais diversas situações. A todos os colegas com quem trabalhei durante estes anos de pesquisa, especialmente aos colegas do GME, pela compreensão e espírito de equipe que contribuíram para a minha experiência prática.

Agradecimento especial ao professor Dr. Altamiro Amadeu Susin pelo incentivo na escolha acertada do assunto que permitiu minha iniciação nesta maravilhosa área que é o processamento digital de imagens, agradeço também ao Fernando Peixoto Coelho de Souza, que contribuiu significativamente para o desenvolvimento deste trabalho, com a implementação da primeira versão do sistema.

Agradeço aos meus pais pela companhia constante, também a minha irmã pelo suporte diário, e ao Mario que durante esta difícil fase em nossas vidas soube compreender e incentivar o desenvolvimento este trabalho.

Enfim, agradeço a Deus por todos os momentos.

Sumário

Lista de Figuras 7

Lista de Tabelas 10

Resumo 11

Abstract 12

1 Introdução 13

1.1 Um Sistema de Análise de Imagem 14
1.2 Objetivos 15
1.3 Estrutura do Trabalho 16

2 Processamento de Imagem 18

2.1 Introdução 18
2.2 Fundamentos de Imagens Digitais 18
2.2.1 Um Modelo Simples de Imagem 18
2.2.2 Aquisição das Imagens 19
2.2.3 Digitalização 20
2.2.4 Tipos de Imagem Digital 21
2.2.5 Amostragem e Quantização 22
2.2.6 Etapas da Análise de Imagens 23
2.3 Tratamento de Imagens 24
2.3.1 Introdução 24
2.3.2 Domínio Espacial e Domínio Freqüência 25
2.3.3 Realce de Imagens 25

3 Reconhecimento de Padrões 39

3.1 Introdução 39
3.2 Redes Neurais 39
3.2.1 Introdução 39
3.2.2 Histórico 41
3.2.3 Modelo de um Neurônio 42
3.2.4 Componentes de uma Rede Neural Artificial 43
3.2.5 Principais Arquiteturas das RNAs 46
3.2.6 Técnicas de Aprendizado 49
3.2.7 Redes Multicamadas e o Algoritmo Backpropagation 49
3.2.8 Modelos de Implementação 51

4 Sistema de Identificação Automática de Veículos 53

4.1 Introdução 53
4.2 Aplicações 54
4.3 Tratamento das Imagens 55
4.4 Sistema Siav 1.0 56
4.4.1 Localização das Placas 56
4.4.2 Extração dos Caracteres 58
4.4.3 Reconhecimento dos Caracteres 58

4.5 Aperfeiçoamento do Sistema 59
4.5.1 Redução da Área de Busca 60
4.5.2 Pré-processamento da Imagem 62
4.5.3 Análise da Tonalidade dos Dígitos 66
4.5.4 Rede Neural 66

5 Resultados 70

5.1 Introdução 70
5.2 Siav 2.0 70
5.3 Estudo Estatístico 71
5.4 Pré-Processamento da Imagem 72
5.4.1 Equalização do Histograma 72
5.4.2 Filtragem Homomórfica 74
5.4.3 Filtro Butterworth. 75
5.5 Análise da Tonalidade dos Dígitos 75
5.6 Rede Neural 76
5.7 Resultados Gerais e Comparativos 80

6 Implementação de Redes Neurais em Hardware 82

6.1 Introdução 82
6.2 Considerações Iniciais 82
6.3 Arquitetura Da Rede Neural 87
6.3.1 Introdução 87
6.3.2 Arquitetura 88
6.3.3 Implementação da Rede Neural 92
6.3.4 Resultados de Síntese 96

7 Conclusões e Perspectivas Futuras 98

7.1 Conclusão 998
7.2 Perspectivas de Trabalhos Futuros 99

Anexo 1 Características dos Componentes 100

Anexo 2 Implementação dos Blocos de Hardware em Fpga para a Rede Neural Siav 2.0 101

I Somador Ripple_Carry 101
II Multiplicador Booth 101
III Gerador de Endereço 104

Anexo 3 Implementação em Vhdl dos Blocos Necessários e da Rede Neural Siav 2.0 105

Bibliografia 117

Lista de Figuras

FIGURA 1.1 - Exemplos dos Bancos de Imagem 14
FIGURA 2.1 - Função δ(x,y) 18
FIGURA 2.2 - Imagem Original e demonstração de parte dos pixels..... 19
FIGURA 2.3 - Etapas da análise e classificação de imagens 23
FIGURA 2.4 - Modelo de um histograma..... 26
FIGURA 2.5 - Imagem de baixo e alto contraste respectivamente 27
FIGURA 2.6 - Imagem Escura..... 28
FIGURA 2.7 - Histograma original..... 28
FIGURA 2.8 - Imagem Clara 28
FIGURA 2.9 - Histograma original..... 29
FIGURA 2.10 - Imagem Escura..... 29
FIGURA 2.11 - Histograma original..... 29
FIGURA 2.12 - Imagem Original 31
FIGURA 2.13 - Histograma original..... 32
FIGURA 2.14 - Imagem Equalizada 32
FIGURA 2.15 - Histograma Equalizado 32
FIGURA 2.16 - Histograma e Threshold de uma Imagem 33
FIGURA 2.17 - Esquema para Binarização 33
FIGURA 2.18 - Imagem Original e Imagem Binarizada 34
FIGURA 2.19 - Histograma da Imagem Binarizada 34
FIGURA 2.20 - Imagem Original 34
FIGURA 2.21 - Imagem Binarizada 35
FIGURA 2.22 - Representação Simbólica de um Filtro 36
FIGURA 2.23 - Relação entre filtro no domínio tempo e no domínio freqüência..... 36
FIGURA 3.1 - Componentes do Neurônio Biológico 42
FIGURA 3.2 - Neurônio de McCulloch e Pitt..... 43
FIGURA 3.3 - Neurônio como unidade limiar..... 44
FIGURA 3.4 - Rede Neural Artificial com uma Camada de Neurônios..... 47
FIGURA 3.5 - Rede Neural Artificial com múltiplas camadas de neurônios 47
FIGURA 3.6 - Rede *Feedforward*..... 48
FIGURA 3.7 - Rede *Feedback* 48
FIGURA 3.8 - Fluxos dos Sinais de entrada e de erro 50
FIGURA 3.9 - Rede Multicamadas Feedforward com treinamento Backpropagation 51
FIGURA 4.8 - Etapas do SIAV 1.0..... 56

FIGURA 4.9 - Caracter 0 (15 x 15) 59
FIGURA 4.10 - Gráfico da Distribuição das Placas em Relação a Média 60
FIGURA 4.11 - Gráfico da Redução da Área de Busca 61
FIGURA 4.12 - Exemplo da área reduzida 61
FIGURA 4.13 - Imagem com baixo contraste (a) e seu histograma (b) 62
FIGURA 4.14 - Imagem Equalizada (a) e seu histograma (b) 63
FIGURA 4.15 - Abordagem da Filtragem Homomórfica 64
FIGURA 4.16 - Imagem Original e Imagem filtrada com filtro Homomórfico 64
FIGURA 4.17 - Imagem Original e Imagem filtrada com Butterworth (8) 65
FIGURA 4.18 - Imagem Original e Imagem filtrada com Butterworth (8) 65
FIGURA 4.19 - Caracteres Extraídos 66
FIGURA 4.20 - (a) Caracteres extraídos sem analise da tonalidade (b) Caracteres extraídos com análise das tonalidades. 66
FIGURA 4.21 - Redes Neurais para Números e Letras 68
FIGURA 5.2 - Imagem Original e Imagem Equalizada pertencente ao Banco1 72
FIGURA 5.3 - Imagem Original e Imagem Equalizada pertencente ao Banco2 73
FIGURA 5.4 - Imagem Original e Imagem Equalizada pertencente ao Banco3 73
FIGURA 5.5 - Processamento da Imagem Original e da Imagem Equalizada respectivamente 73
FIGURA 5.6 – Processamento da Imagem Original e da Imagem Filtrada respectivamente 74
FIGURA 5.7 – Imagem Original e Imagem Filtrada pelo filtro Butterworth de ordem n 75
FIGURA 5.8 – Resultado da Etapa de Extração 76
FIGURA 5.9 – Resultado da Etapa de Extração do SIAV1 e do SIAV2 76
FIGURA 5.10 – Rede para reconhecimento dos números e letras respectivamente 77
FIGURA 5.11 – Exemplos de extração de caracteres 78
FIGURA 5.12 – Função de Ativação Logística 78
FIGURA 5.13 – Função de Ativação Tangente Hiperbólica 79
FIGURA 5.14 – Resultados Comparativos das Redes Neurais 80
FIGURA 5.15 – Resultados Comparativos no SIAV 1.0 80
FIGURA 5.16 – Resultados Comparativos no SIAV 2.0 81
FIGURA 5.17 – Resultados Comparativos no SeeCar 81
FIGURA 6.4 – Espaços de endereçamento das memórias da rede 88
FIGURA 6.5 – Circuito de endereçamento e organização da memória da rede 89
FIGURA 6.6 – Estrutura da parte operativa da rede 90
FIGURA 6.7 – Estrutura da parte de controle da rede 91
FIGURA 6.8 – I/O da Rede Neural Reduzida para Reconhecimento de Números 92
FIGURA 6.9 – Espaço de Endereçamento das Memórias da Rede 93
FIGURA 6.10 – Circuito de endereçamento e organização da memória da rede 94

FIGURA 6.11 – Estrutura da Parte Operativa da Rede Reduzida para reconhecimento dos números 95
FIGURA 6.12 – Diagrama de Estados da Parte de Controle da Rede Reduzida para reconhecimento dos números 96
FIGURA B1 – Dados da simulação do multiplicador..... 101
FIGURA B2 – Multiplicação de números de 12 bits com sinal pelo algoritmo de Booth..... 102
FIGURA B3 – Diagrama da Máquina de estados do multiplicador tipo Booth..... 103
FIGURA B4 – Estrutura do Multiplicador Booth..... 103
FIGURA B5 – Simulação do Multiplicador..... 103

Lista de Tabelas

TABELA 2.1 - Dispositivos de Entrada do Sistema 20
TABELA 3.1 - Diferenças Entre o Computador e o Cérebro Humano [Molz98] 40
TABELA 4.2 - Associação dos Neurônios da Camada de Saída 69
TABELA 6.1 - Tabela da Função Tangente Hiperbólica 84
TABELA 6.2 - Tabela dos Pesos da Camada Escondida 86
TABELA 6.3 - Tabela dos Pesos da Camada de Saída para a Rede de Letras 86
TABELA 6.4 - Capacidade de Memória Necessária para a Rede para Reconhecimento de Letras 87
TABELA 6.5 - Capacidade de Memória Necessária para a Rede para o Reconhecimento dos Números 87
TABELA 6.6 - Capacidade de Memória Necessária para a Rede Reduzida para o Reconhecimento dos Números 93
TABELA 6.7 – Resultado de Síntese dos Componentes Externos à Rede 97
TABELA 6.8 - Resultado de Síntese da Rede Neural 97
TABELA 6.9 - Resultado de Desempenho da Rede Neural 97
TABELA A1 - Tabela Dados Apex20k 100
TABELA B1 – Dados da Simulação do Somador 101
TABELA B2 – Ações Sobre o Produto Parcial para o Algoritmo de Booth 102
TABELA B3 – Dados da Simulação do Muliplicador 103
TABELA B4 – Dados da Simulação do Gerador 104

Resumo

Sistemas de visão artificial são cada vez mais usados para auxiliar seres humanos a realizar diferentes tarefas. Estes sistemas são capazes de reconhecer padrões em imagens complexas.

Técnicas de visão computacional têm encontrado crescente aplicação em estudos e sistemas de controle e monitoração de tráfego de automóveis. Uma das áreas de pesquisa que tem sido objeto de estudo por diferentes grupos é a leitura automática de placas de matrículas como forma de detectar transgressores, encontrar carros roubados ou efetuar estudos de origem/destino [BAR99].

Com o constante crescimento do volume de tráfego de automóvel e a limitada capacidade dos sensores convencionais, especialistas da área recorrem a técnicas de identificação automática de veículos para obter dados relativos ao escoamento de tráfego.

A identificação automática de veículos tem tido essencialmente duas abordagens distintas: a utilização de *transponders* e a utilização de técnicas de visão computacional [INI85] . Estas são essencialmente úteis em casos em que não é viável obrigar os motoristas a instalar *transponders* em seus automóveis. No entanto, essas técnicas são mais sensíveis às condições atmosféricas e de iluminação tais como nevoeiros, chuva intensa, luz noturna, reflexos em superfícies, etc.

Este trabalho apresenta um estudo de diversas técnicas de processamento de imagem objetivando o aperfeiçoamento de um sistema de identificação automática de placas de veículos. Este aperfeiçoamento está relacionado com a diminuição do tempo de execução necessário à localização e reconhecimento dos caracteres contidos nas placas dos veículos bem como a melhorar a taxa de sucesso no seu reconhecimento.

A primeira versão do sistema de identificação da placas de veículos descrito em [SOU2000], desenvolvido no CPG-EE da UFRGS, denominado SIAV 1.0, localiza e extrai 91,3% das placas corretamente mas apresenta uma taxa de reconhecimento das placas de 37,3%, assim como um tempo de processamento não satisfatório.

Neste trabalho, cujo sistema desenvolvido é denominado SIAV 2.0, a imagem é previamente processada através da aplicação de técnicas de realce da imagem. O principal objetivo das técnicas de realce é processar a imagem de modo que o resultado seja mais apropriado para uma aplicação específica do que a imagem original [GON93]. O sistema busca melhorar a qualidade da imagem eliminando ou suavizando sombras e reflexos presentes na cena em virtude da iluminação não controlada. Visando um menor tempo de execução durante o tratamento e análise da imagem um estudo estatístico baseado na distribuição gaussiana foi realizado de maneira a restringir a área de análise a ser processada.

O SIAV possui duas redes neurais como ferramentas de reconhecimento de caracteres. A partir da análise dos diferentes modelos de redes neurais empregados na atualidade, foi desenvolvida uma nova arquitetura de rede a ser utilizada pelo SIAV 2.0 que oferece uma taxa de reconhecimento superior a rede neural usada no SIAV 1.0. Visando um melhor tempo de execução, a implementação em hardware dedicado para este modelo é abordado.

Os testes foram realizados com três bancos de imagens obtidas por câmeras diferentes, inclusive por dispositivo "pardal" comercial. Estes testes foram realizados para verificar a efetividade dos algoritmos aperfeiçoados.

Palavras-chaves: Identificação Automática de Veículos, Processamento de imagem, Redes neurais.

TITLE: “IMPROVING THE TECHNIQUES FOR VEHICLE PLATE CHARACTER RECOGNITION”

Abstract

Artificial vision systems are used more and more to aid human beings in different tasks. These systems are capable to recognize patterns previously taught in a complex image.

Techniques of computational vision have found increasing application in studies and systems of control of traffic of automobiles. One of the search areas that have been object is the automatic identification of vehicle license plates as form to detect transgressors, to find steal car or to effect studies of origin /destiny [BAR99]. With the constant growth of the volume of automobile traffic and the limited capacity of the conventional device, specialists of the area appeal the techniques of automatic identification of vehicles to get data relative to the draining of traffic.

The automatic identification of vehicles has had two distinct boarding essentially: the use of transponders and the use of techniques of computational vision [INI85] . These are useful in cases where it is not viable to compel the drivers to install transponders in its automobiles, but these techniques are more sensitible to the atmospheric conditions and of illumination.

This work presents a study of diverse techniques of image processing objectifying the improvement for a system of automatic identification of license plates of the vehicles. This improvement is related with the reduction of the time of execution to the location and recognition of the characters contained in the plates of the vehicles as well as improving the tax of success in its recognition.

The first version of the identification system for the license plates of vehicles described in [SOU2000], developed in the CPG-EE of UFRGS, called SIAV 1.0, presents a good tax of success in the location of the boards and correct extraction of the characters, but it presents low tax in the recognition of the plates as well as a time of processing not satisfactory.

In this work, whose developed system is called SIAV 2.0. The image is previously processed through the application of techniques of enhance of the image. The main objective of the techniques of enhance is to process the image of mode that the result is more appropriate for a specific application of that the original picture [GON93], eliminating or alleviating shades and reflection in the scene virtue of the bad illumination. Surching a lesser time of execution during the handling and analysis of the picture a statistical study based was carried through in way to restrict the analysis area to be processed.

The SIAV possess two neural networks as tools of recognition of characters. These networks had alterations. From the analysis of the different models of neural networks a new architecture of network was developed to be used by SIAV 2.0 that offers a tax of recognition upper the neural network used in SIAV 1.0. Aiming one better execution time, the implementation in the dedicated hardware for this model is boarded.

Three distinct sets of images will be used to evaluate the performance of the system. The tests was taken with three banks of pictures gotten for elements of captures different, also for device known systems as "pardais", being constituted of pictures of automobiles infractors, captured in the transit of the city of Porto Alegre with vehicles in movement and excess of speed in public ways.

Keywords: Image Processing, Neural network, Autimatic Identification of Vehicle

1 Introdução

A representação e processamento da informação visual tem cumprido um papel fundamental na vida do ser humano desde seus primórdios, como é atestado pelas pinturas pré-históricas. Desde então, as imagens têm cumprido diferentes papéis no dia a dia das pessoas em praticamente todo o mundo, tanto na representação gráfica que auxilia o entendimento e o registro para posterior recordação, como em aplicações práticas, incluindo a automação de tarefas repetitivas e/ou perigosas.

O processamento digital de imagens é relativamente recente em relação à fascinação humana por estímulos visuais [CAS96].

A área de processamento de imagens teve início em meados da década de 60. O aumento da capacidade de memória e da velocidade dos computadores contribuiu para dar impulso à nova tecnologia. Esta é uma área interdisciplinar, utilizando conceitos da informática, física e eletrônica, entre outras, e está adquirindo uma importância cada vez maior pois é útil em diversas outras áreas do conhecimento [LIB97] .

A vontade e a necessidade crescente de automatização das atividades cotidianas evidenciam um envolvimento crescente das ferramentas de processamento de imagens em um grande número de domínios. Pode-se citar algumas aplicações como tarefas industriais, reconhecimento de padrões e reconstrução tridimensional.

Considerável esforço tem sido dedicado à solução do problema de reconhecimento e caracterização de objetos presentes em uma imagem. Até a década de 70, predominou o uso de técnicas ópticas de processamento. A partir do início dos anos 80, com os avanços da microeletrônica e o desenvolvimento de arquiteturas paralelas de processamento, as técnicas digitais passaram a ser mais empregadas. Atualmente, o amadurecimento das técnicas computacionais inteligentes, como sistemas especialistas, lógica nebulosa, redes neurais e algoritmos genéticos, têm permitido novas abordagens para o problema de reconhecimento de padrões.

A complexidade do problema de reconhecimento e de classificação de imagem, que dificilmente pode ser abordado em termos algorítmicos, tem tornado o uso dessas técnicas cada vez mais freqüentes, especialmente as redes neurais artificiais.

Redes neurais são sistemas paralelos distribuídos, compostos por unidades de processamento simples que computam determinadas funções matemáticas [HAY94]. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos estas

conexões representam valores os quais são associados certos pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar as entradas recebidas por cada neurônio da rede. O funcionamento das redes neurais é inspirado em uma estrutura física concebida pela natureza: o cérebro humano.

1.1 Um Sistema de Análise de Imagem

O processamento de imagens digitais abrange uma ampla gama de hardware, software e fundamentos teóricos [INI85] .

A aplicação à qual este sistema está voltado é a utilização de técnicas de processamento de imagem para leitura automática de placas de veículos. A Figura 1.1 demonstra que o objetivo global é produzir um resultado a partir do domínio do problema por meio de processamento de imagem.

O domínio do problema consiste em placas de veículos e o objetivo é ler e reconhecer o conteúdo de cada uma delas. Assim, a saída desejada é uma seqüência de códigos de caracteres alfanuméricos correspondentes à matrícula do veículo.

FIGURA 1.1 - Exemplos dos Bancos de Imagem

O primeiro passo no processo é a aquisição da imagem, para isso necessita-se de um sensor para imageamento e a capacidade de digitalizar o sinal produzido pelo sensor. Após a obtenção da imagem é necessário realizar o pré-processamento. A função do pré-processamento é melhorar a imagem de forma a aumentar as chances de sucesso do processamento computacional que se faz posteriormente sobre a imagem. O próximo estágio trata da segmentação. No caso do reconhecimento de caracteres, o papel básico da segmentação é extrair caracteres individuais do fundo da imagem.

A saída do estágio de segmentação é constituído tipicamente por dados em forma de *pixels*, correspondendo tanto à fronteira de uma região como a todos os pontos dentro da mesma. É necessário converter os dados para uma forma adequada ao processamento computacional através da representação e descrição.

O último estágio envolve reconhecimento e interpretação. Reconhecimento é o processo que atribui um rótulo a um objeto, baseado na informação fornecida pelo seu descritor. A interpretação envolve a atribuição de significado a um conjunto de objetos reconhecidos.

1.2 Objetivos

Este trabalho descreve um sistema de reconhecimento de caracteres alfanuméricos aplicado à identificação de placas de veículos e tem como alvo o aperfeiçoamento deste sistema.

Os objetivos deste trabalho foram (a) aperfeiçoar o software SIAV1.0, demonstrando que as técnicas a serem implementadas no SIAV2.0 mesmo que já conhecidas na literatura, resultam em melhores taxas de reconhecimento de placas e (b) aperfeiçoar o software SIAV1.0 de forma a reduzir o tempo de processamento requerido para o reconhecimento.

Atualmente o processo de reconhecimento das placas dos veículos é realizado de forma manual por pessoas que analisam as imagens. Normalmente, em um único ponto da cidade de Porto Alegre - RS, um *pardal* aplica acima de mil multas em um dia de funcionamento. Esta é uma das aplicações onde pesquisar uma solução automática para o problema de reconhecimento torna-se importante também do ponto de vista prático.

O sistema projetado visa reconhecer os caracteres presentes na placa de um veículo segundo a atual legislação brasileira de trânsito, com sete caracteres (3 letras e 4 números), fundo claro e caracteres escuros e uma distribuição de luz não homogênea sobre a superfície da placa.

Três bancos distintos de imagens foram utilizados para avaliar o desempenho do sistema. Estes bancos são compostos por imagens reais, adquiridas em situações cotidianas tais como no estacionamento, no trânsito livre, nas ruas.

O primeiro banco é composto por 250 imagens de automóveis estacionados nas ruas da cidade de Porto Alegre. O segundo banco, também com 250 imagens, contém imagens de automóveis em movimento, capturadas por câmera fotográfica operada manualmente por um observador. O terceiro e último banco, também com 250 imagens,

foi fornecido pela Companhia de Processamento de Dados – PROCEMPA, e é constituído de imagens de automóveis infratores, capturados automaticamente por equipamentos apelidados de *pardais*, pertencentes ao banco de dados desta empresa.

1.3 Estrutura do Trabalho

O texto apresentado está dividido em sete capítulos. O Capítulo 1 faz uma introdução do assunto, dando ao leitor uma visão geral do trabalho desenvolvido.

No Capítulo 2, apresenta-se mais detalhes da área de processamento de imagens. Esta área é bastante ampla. Foi dado destaque aos tópicos de maior relevância para a compreensão do trabalho, como é o caso dos algoritmos voltados ao tratamento de imagens, principalmente técnicas de realce e filtragem da imagem.

No Capítulo 3, é feita uma breve descrição da área de redes neurais para que o leitor sem experiência nesta área possa compreender como e por que essa técnica foi utilizada no sistema desenvolvido. Da mesma forma que a área de processamento de imagens, essa área também é muito ampla e procurou-se direcionar o assunto para os tópicos de maior relevância. A rede neural *feedfoward* com o algoritmo de treinamento *backpropagation* é abordada com mais detalhe, pois esse modelo é utilizado no sistema.

A partir do Capítulo 4 são apresentadas as técnicas utilizadas neste trabalho, descrevendo inicialmente o funcionamento do sistema de identificação automática de veículos, denominado SIAV 1.0, que serviu como ponto de partida para este trabalho. A seguir o sistema desenvolvido é descrito, com ênfase nas novas técnicas e as alterações realizadas em busca de uma melhor taxa (ou velocidade) de processamento e uma taxa de reconhecimento dos sete caracteres presentes nas placas superior à versão anterior. Esta nova versão do sistema foi denominada SIAV 2.0 e escrita em linguagem C com algumas simulações realizadas também no Matlab para testar algoritmos e alternativas de melhorias.

O Capítulo 5 apresenta uma proposta de arquitetura para a nova rede neural desenvolvida.

Os resultados obtidos com os testes do sistema desenvolvido e sua análise estão descritos no Capítulo 6. O sistema foi analisado utilizando 750 imagens de automóveis distribuídas em três bancos distintos, o que corresponde a 5.250 caracteres. Os resultados obtidos foram comparados aos do SIAV 1.0 e também aos resultados obtidos por um software demonstrativo israelense - denominado SeeCar - disponível no site

http://www.htsol.com da empresa israelense High Tech Solutions e utilizado como parâmetro de comparação no SIAV 1.0 [SOU2000].

As conclusões e as perspectivas futuras para a continuidade deste trabalho são citadas no Capítulo 7.

2 Processamento de Imagem

2.1 Introdução

Neste capítulo serão revisados alguns conceitos fundamentais para o entendimento do que será apresentado a seguir.

Inicialmente serão abordados os fundamentos das imagens digitais como a definição do que é imagem digital e suas características. A seguir serão abordadas técnicas voltadas ao tratamento de imagens tanto no domínio espacial quanto no domínio freqüência.

2.2 Fundamentos de Imagens Digitais

2.2.1 Um Modelo Simples de Imagem

A palavra imagem é atribuída a uma função bidimensional *f(x,y)* contínua, onde para qualquer par *(x,y)* existe um valor *f* proporcional a intensidade do brilho da imagem naquele ponto. As coordenadas espaciais *(x,y)* localizam qualquer ponto pertencente a imagem em questão [SOU2000].

Como os computadores não manipulam dados analógicos, é necessário converter uma imagem contínua em sua forma digital. Teoricamente, isto pode ser feito através da multiplicação da imagem contínua *f(x,y)* por uma função delta de Dirac bidimensional δ(x,y), Figura 2.1.

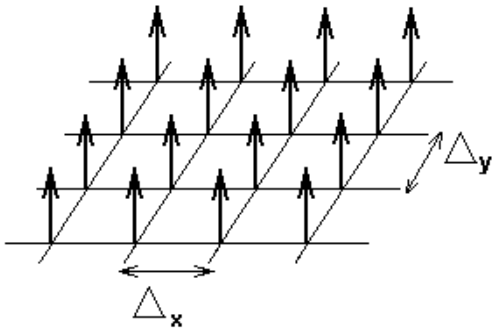


FIGURA 2.1 - Função δ(x,y)

O termo imagem digital refere-se a uma imagem que pode ser discretizada quanto as suas coordenadas espaciais e quanto a intensidade de seu brilho[GON93]. Uma imagem digital pode, então, ser considerada uma matriz cujos índices das linhas e colunas identificam um ponto dentro da imagem e o correspondente valor do elemento

da matriz identifica o nível de cinza naquele ponto. Os elementos desta matriz digital são chamados de elementos da imagem, elementos da figura, "pixels" ou "pels", estes dois últimos, abreviação de picture elements [GON93].

Imagens digitais são formadas por pixels como pode ser observado na Figura 2.2.

FIGURA 2.2 - Imagem Original e demonstração de parte dos pixels

As imagens que as pessoas percebem em atividades visuais corriqueiras são formadas por luz emitida ou refletida pelos objetos. A natureza básica de f(x, y) pode ser caracterizada por dois componentes, a quantidade de luz de uma fonte luminosa incidindo na cena sendo observada e também pela quantidade de luz refletida pelos objetos da cena. Apropriadamente, esses componentes são chamados iluminação e reflectância, respectivamente, e são representados por i(x, y) e r(x, y). O produto destas funções resulta em f(x, y).

$$f(x, y) = i(x, y) \cdot r(x, y) \text{ onde}$$

$$0 < i(x, y) < \alpha$$

$$0 < r(x, y) < 1$$

A última equação indica que a reflectância é limitada entre 0 (absorção total) e 1 (reflectância total). A natureza de $i(x, y)$ é determinada pela fonte de luz e $r(x, y)$ é determinada pelas características dos objetos presentes na cena.

2.2.2 Aquisição das Imagens

Dois elementos são necessários para a aquisição de imagens digitais. O primeiro é um dispositivo físico que seja sensível a uma banda do espectro de energia eletromagnética e que produza um sinal elétrico de saída proporcional a um nível de energia detectada. O segundo, chamado *digitalizador*, é um dispositivo para a conversão da saída elétrica de um dispositivo de sensoriamento físico para a forma digital [GON93].

O processo de aquisição de imagens digitais, consiste em transformar as imagens reais em imagens digitais. De acordo com o destino da imagem, os dispositivos de entrada podem ser classificados como vetoriais ou matriciais[GON93].

Os dispositivos de entradas vetoriais são em sua maioria utilizados em sistemas interativos, onde o usuário tem uma participação direta com estes dispositivos. Como existem várias maneiras de interagir com as máquinas, tem-se alguns exemplos de dispositivos vetoriais: o light pen", a "tablet", o "touch screen", o "3D-digitizer".

Os dispositivos de entrada matricial são em geral utilizados de forma não-interativa, desta forma, adequando-se mais à aquisição de grandes volumes de dados.

Dentro do ambiente do sistema desenvolvido as imagens são capturadas por diferentes dispositivos e a aquisição destas é feita em cores. Como dispositivo de entrada para as imagens do ambiente, para o primeiro banco de testes, tem-se uma câmera CCD que capta imagens com 320 x 240 pixels e com representação das cores em 24 bits/pixel (*true color*). O dispositivo para o segundo banco de testes é uma câmera fotográfica digital. O terceiro banco de imagens foi capturado por pardais distribuídos pelas ruas de POA e fornecido pela PROCEMPA.

Os dispositivos de entrada para os diferentes bancos de dados possuem as seguintes características:

TABELA 2.1 - Dispositivos de Entrada do Sistema

Dispositivo	*Resolução óptica*	*Resolução de cor*	*Imagens*
Câmera CCD	320X240 dpi	24 bits/pixel	250 Imagens
Máquina Fotográfica Digital	1024X1024 dpi	24 bits/pixel	250 Imagens
Pardais	540X480 dpi	24 bits/pixel	250 Imagens

2.2.3 Digitalização

Como os computadores podem processar apenas imagens codificadas em informações digitais, e as imagens na natureza encontram-se em outras formas, um pré-requisito para o processamento digital de imagens é a conversão de uma imagem para a forma digital [CAS96], a esta conversão dá-se o nome de digitalização de imagens.

O processo de digitalização consiste em realizar a aquisição de uma imagem, a qual é passada para o computador em um formato adequado para que este possa processá-la. As informações visuais são convertidas em sinais elétricos, e estes sinais são quantificados em valores binários e armazenados na memória do computador. No processo de digitalização, os sinais são amostrados espacialmente e quantificados em amplitude, de forma a obter a imagem digital.

Um digitalizador de imagens deve ter a capacidade de dividir uma imagem em *pixel* e endereçá-los individualmente para: medir a quantidade de energia em cada pixel, quantificar a medição contínua para produzir um conjunto de valores inteiros, e escrever este conjunto em um dispositivo de armazenamento de dados [CAS96].

Para se realizar uma amostragem de uma imagem deve ser fixado um intervalo que vai determinar a freqüência de amostragem e consequentemente o número de pontos discretizados (resolução da imagem). É assumido que o intervalo de amostragem entre duas células da imagem seja ao menos igual à metade do menor intervalo significativo da cena capturada, ou seja, a freqüência espacial deve ser o dobro da freqüência espacial necessária para visualizar o menor elemento significativo presente na cena [PAZ88]. No caso da digitalização utilizando-se uma câmera, a distância do objeto alvo até a câmera será importante na sua captura, pois esta distância influi diretamente no tamanho da imagem do objeto. O tamanho da superfície de projeção e as características do sistema óptico são responsáveis pelo desempenho do sistema.

No processo de digitalização, a imagem sofrerá uma amostragem espacial, conforme citado anteriormente, e uma discretização da intensidade luminosa, que é denominada de quantização. No processo de quantização, uma imagem com tons contínuos é convertida em uma de tons discretos. Para o armazenamento e processamento, cada tonalidade (intensidade da luz refletida por cada ponto da imagem) é representada por um valor armazenado de forma binária. Cada ponto amostrado possuirá portanto um valor binário correspondente à intensidade luminosa da imagem naquele ponto.

2.2.4 Tipos de Imagem Digital

Conforme foi visto, imagens digitais são formadas por um conjunto de *pixels*. O *pixel* é o menor elemento da imagem e possui um nível de cinza associado a cada um.

2.2.4.1 Imagem Binária

São imagens onde os *pixels* assumem os valores 0 (preto) ou 1 (branco), assim necessitam de apenas 1 bit por *pixel* para serem representados. São utilizadas normalmente para destacar os objetos de interesse, já que a imagem binária é a mais desprovida de detalhes.

2.2.4.2 Imagem Monocromática

São imagens onde os *pixels* podem assumir valores entre 0 e *N*, representando a intensidade do cinza onde 0 significa intensidade nula (preto) e *N* a intensidade máxima

(branco), todos os outros valores intermediários serão tons de cinza. Geralmente, devido à representação em 8 bits por pixel, a intensidade em um ponto qualquer pode assumir 256 valores diferentes , de 0 a 255, representando os 256 tons diferentes.

2.2.4.3 Imagem Colorida

A mais familiar forma de imagem multi-espectral é a colorida. Estas imagens são usualmente compostas por um conjunto de 24 bits, 8 bits para representar as intensidades de vermelho, 8 bits para o verde e 8 bits para o azul. Com a combinação dessas três cores básicas, utilizando-se 24 *bits/pixel*, pode-se chegar a um número de até 16 milhões de cores e tonalidades distintas. Este número é perfeitamente adequado para a representação da realidade sem perda de detalhes e qualidade em relação à cores, pois está acima da capacidade do olho humano em distinguir cores e tonalidades.

2.2.5 Amostragem e Quantização

Para ser adequada para processamento computacional, uma função *f(x, y)* precisa ser digitalizada tanto espacialmente quanto em amplitude. A digitalização das coordenadas espaciais *(x, y)* é denominada amostragem da imagem e a digitalização da amplitude é chamada quantização em níveis de cinza [GON93].

O processo de amostragem pode ser compreendido como a partição do plano *xy* em uma grade, com as coordenadas de cada cruzamento da grade sendo um par de elementos obtidos do produto cartesiano Z^2, que é o conjunto de todos os pares ordenados *(a,b)*, com a e b sendo elementos de Z.

Para realizar a amostragem de uma imagem deve ser fixado um intervalo de tempo que vai determinar a freqüência de amostragem e consequentemente o número de pontos discretizados (resolução da imagem). É assumido na prática que o intervalo de amostragem entre duas células da imagem seja, no mínimo, igual à metade do menor intervalo significativo da cena capturada, ou seja, a resolução deve ser o dobro da resolução necessária para visualizar o menor elemento significativo presente na cena [PAZ88], ainda que na prática se utilize resoluções da ordem de dez vezes a resolução mínima necessária para que não haja perda de informação.

No processo de quantização, uma imagem com tons contínuos é convertida em uma imagem de tons discretos. Para o armazenamento e processamento por um computador, cada tonalidade é representada por um valor binário armazenado. Cada ponto amostrado possuirá portanto um valor correspondente à intensidade luminosa da imagem naquele ponto.

2.2.6 Etapas da Análise de Imagens

A análise de imagens de uma maneira geral é executada em um determinado número de etapas, onde são realizadas funções específicas. Pode-se dividir o processamento de imagens de acordo com a figura 2.3, onde estão presentes as etapas de aquisição, o pré-processamento, a segmentação, representação e descrição e por fim as etapas de reconhecimento e interpretação da imagem.

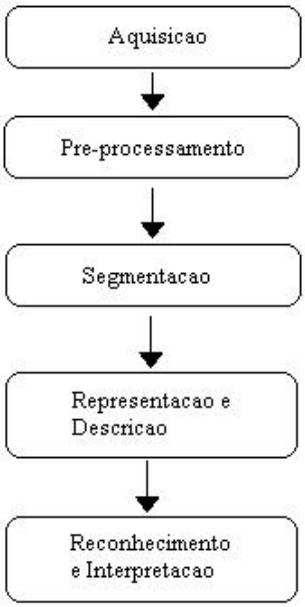


FIGURA 2.3 - Etapas da análise e classificação de imagens

A análise de imagens tem como objetivo descrever uma dada imagem. Esta descrição é dependente do domínio da aplicação [ROS84].

2.2.6.1 Aquisição

O primeiro estágio da análise de imagens consiste no processo de aquisição. Durante esta etapa, sensores ópticos são responsáveis pela captura de sinais. Paralelamente, estes sinais são digitalizados por conversores AD, e adquiridos por sistemas de entrada do equipamento computacional.

2.2.6.2 Pré-processamento

Geralmente, durante a fase de aquisição de imagens, além da informação de interesse, são capturados sinais espúrios que não possuem significado e acabam prejudicando a análise da imagem. Tais sinais são conhecidos por ruídos. Um dos objetivos da fase de pré-processamento é justamente a remoção, ou ao menos a diminuição da influência dos ruídos sobre a imagem. Os dados também podem sofrer

conversões, escalonamentos e diversas outras formas de manipulação, de acordo com a necessidade do sistema como um todo.

Assim, o pré-processamento objetiva o melhoramento da imagem digital, buscando a melhor representação possível para uma dada tarefa.

2.2.6.3 Segmentação

O processo de segmentação pode ser entendido como o particionamento de uma imagem em regiões que apresentem propriedades semelhantes, como textura ou cor. O princípio da segmentação foi apresentado por psicólogos alemães, quando foi mostrado que o ser humano, no processo de visão, realiza naturalmente o agrupamento de regiões por critério baseados na proximidade, similaridade e continuidade [FAC93].

2.2.6.4 Representação e Descrição

Uma vez que uma imagem tenha sido segmentada, os agrupamentos resultantes de pixels segmentados são usualmente representados e descritos em um dado formato para o processamento subsequente [GON93]. Basicamente, a representação de uma região envolve duas escolhas: representá-la em termos de suas características externas ou em termos de suas características internas. A próxima tarefa é descrever a região baseado na representação escolhida.

2.2.6.5 Reconhecimento e Interpretação

Por fim, as etapas de reconhecimento e interpretação que podem ser consideradas etapas de alto nível [GON93]. Esses dois processos possuem uma forte semelhança com aquilo que é geralmente requerido como *cognição inteligente*. A maioria das técnicas das etapas anteriores incluem um conjunto razoavelmente bem definido de formulações teóricas. Entretanto, na etapa de reconhecimento e interpretação o conhecimento e a compreensão dos princípios fundamentais tornam-se menos precisos e mais especulativos [GON93]. Assim, o produto final é um sistema com capacidades operacionais altamente especializadas.

2.3 Tratamento de Imagens

2.3.1 Introdução

As imagens digitais podem sofrer modificações, através de um tratamento visando a alteração de suas características, tanto de resolução quanto de quantização.

Tratar uma imagem consiste em transformá-la sucessivamente afim de deixar mais acessível o seu conteúdo. Existem várias técnicas de processamento de imagens

que dividem-se em duas grandes categorias: métodos no domínio espacial e métodos no domínio freqüência.

2.3.2 Domínio Espacial e Domínio Freqüência

O termo *domínio espacial* refere-se ao agregado de pixels que compõem uma imagem, e métodos no domínio espacial são procedimentos que operam diretamente sobre estes pixels [GON93]. O domínio espacial refere-se ao próprio plano da imagem.

As operações no domínio espacial são realizadas diretamente com os pixels da imagem, o que é uma vantagem, pois a imagem não sofre transformações prévias e posteriores para poder ser processada.

Funções de processamento de imagens no domínio espacial podem ser expressas como:

$$g(x, y) = T[f(x, y)]$$

onde $f(x, y)$ é imagem de entrada, $g(x, y)$ é a imagem processada e T é um operador sobre f, definido sobre alguma vizinhança de (x, y).

Técnicas de processamento no domínio da freqüência são baseadas na utilização das transformadas de Fourier das imagens.

Um conceito importante no domínio da freqüência é o teorema da convolução. Seja $g(x, y)$ uma imagem formada pela convolução de uma imagem $f(x, y)$ e um operador linear invariante com a posição $h(x, y)$, isto é,

$$g(x, y) = h(x, y) * f(x, y).$$

Então, do teorema da convolução, a seguinte relação no domínio da freqüência é verificada: $G(u, v) = H(u, v) \cdot F(u, v)$, em que G, H e F são as transformadas de Fourier de g, h e f respectivamente.

2.3.3 Realce de Imagens

A técnica de realce de contraste tem por objetivo melhorar a qualidade das imagens sob os critérios subjetivos do olho humano. É normalmente utilizada como uma etapa de pré-processamento para sistemas de reconhecimento de padrões [GON93].

O objetivo principal das técnicas de realce é processar uma imagem, de modo que o resultado seja mais apropriado para uma aplicação específica do que a imagem original. O contraste entre dois objetos pode ser definido como a razão entre os seus níveis de cinza médios.

A manipulação do contraste consiste numa transformação radiométrica em cada pixel, com o objetivo de aumentar a discriminação visual entre os objetos presentes na imagem.

2.3.3.1 Processamento Ponto a Ponto

2.3.3.1.1 Processamento do Histograma

O histograma é uma das formas mais comuns de se representar a distribuição dos níveis de cinza de uma imagem. Ele fornece a informação de quantos pixels na imagem possuem um determinado nível de cinza, definido dentro de um domínio, por exemplo entre 0 (preto) e 255 (branco). Outra característica é que o histograma não apresenta nenhuma informação espacial da imagem e sim uma função de probabilidade de encontrar um nível de cinza referente a um objeto qualquer da imagem. Normalmente tem-se no eixo X a distribuição dos níveis de cinza e no eixo Y a freqüência em que ocorrem em uma dada imagem, como mostrado na figura 2.4.

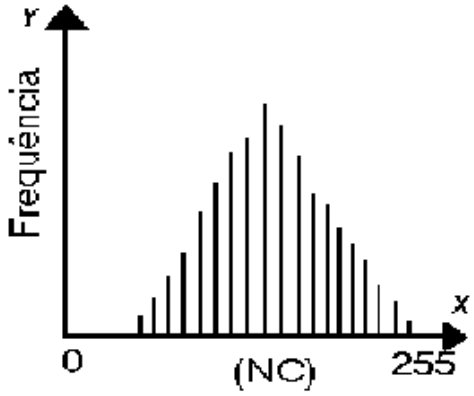


FIGURA 2.4 - Modelo de um histograma

Um histograma descreve a distribuição estatística dos níveis de cinza em termos do número de amostras com cada nível. Esta distribuição pode também ser dada em termos da percentagem do número total de pixels na imagem. Pode ser estabelecida uma analogia entre o histograma de uma imagem e a função densidade de probabilidade, que é um modelo matemático da distribuição de tons de cinza de uma classe de imagens.

A forma do histograma fornece informações importantes como a intensidade média e espalhamento dos valores de níveis de cinza, sendo este último a medida de contraste da imagem. Quanto maior o espalhamento ao longo do eixo dos níveis de cinza, maior o contraste da imagem. A figura 2.5 ilustra esta distribuição dos níveis de cinza.

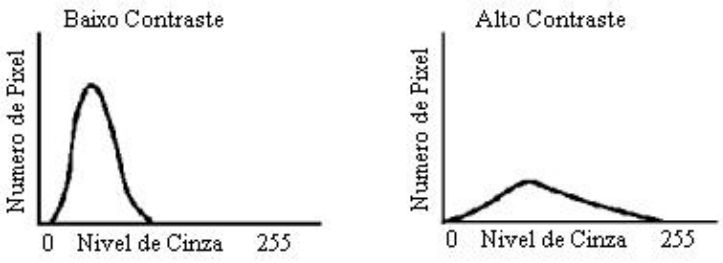


FIGURA 2.5 - Imagem de baixo e alto contraste respectivamente

O histograma normalizado de uma imagem digital com níveis de cinza no intervalo de [0, L-1] é uma função discreta do tipo:

$$p(r_k) = n_k / n$$

onde r_k é o k-ésimo nível de cinza,

n_k é o número de pixels na imagem com este nível de cinza,

n é o número total de pixels na imagem e

k ordem de um determinado nível de cinza (0, 1, ..., L-1).

Assim, $p(r_k)$ é uma probabilidade de ocorrência do nível de cinza r_k.

Um pseudocódigo, em linguagem C, é apresentado para demonstrar o funcionamento do algoritmo que calcula os valores n_k do histograma. A série n_0, n_1, ..., n_{k-1} é uma contagem que designamos por histograma desnormalizado.

```
char imagem[linhas][colunas];       // imagem de tamanho linhas por colunas
int histograma[256];                // vetor do histograma
int linha, coluna, i;
    for (i = 0; i < 256; i++)       // zera inicialmente o vetor de historgama
        histograma[ j ] = 0;
    for (linha = 0; linha < linhas; linha++)
        for(coluna = 0; coluna < colunas; coluna++)
            histograma [(int) imagem [linha][coluna]] ++;   // de 0 a 255
```

Um gráfico desta função discreta e inteira, como histograma desnormalizado ($p(r_k).n$) para todos os valores de k fornece uma descrição global da aparência de uma imagem.

O histograma desnormalizado mostrado na figura 2.7 representa a contagem de ocorrências dos níveis de cinza contidas na imagem da figura 2.6, e mostra que os níveis de cinza estão concentrados em direção à extremidade escura. Assim, este histograma corresponde a uma imagem com características predominantemente escuras.

FIGURA 2.6 - Imagem Escura

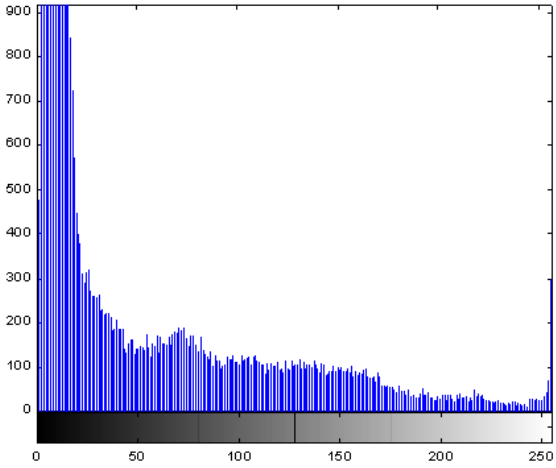


FIGURA 2.7 - Histograma original

Exatamente o oposto acontece na figura 2.9 que representa os níveis de cinza da imagem contida na figura 2.8, onde tem-se uma imagem com características predominantemente claras.

FIGURA 2.8 - Imagem Clara

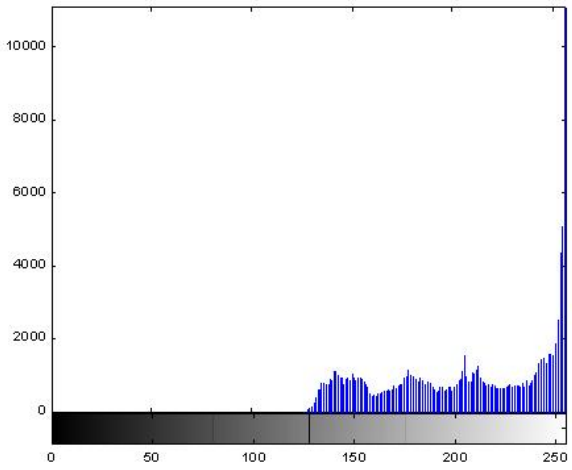


FIGURA 2.9 - Histograma original

Finalmente, a figura 2.11 mostra um histograma com espalhamento significativo, corresponde a uma imagem de alto contraste.

FIGURA 2.10 - Imagem Escura

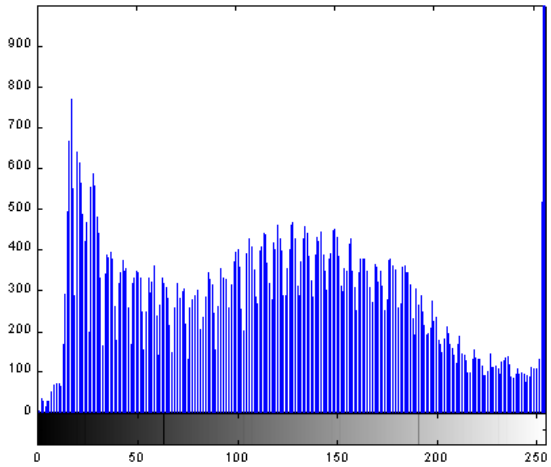


FIGURA 2.11 - Histograma original

Embora as propriedades acima discutidas sejam descrições globais que não dizem nada específico sobre o conteúdo da imagem, a forma do histograma de uma imagem nos dá informação útil sobre a possibilidade para realce do contraste.

2.3.3.1.2 Modificação da Escala de Cinza

Dentro desta categoria de operação destacam-se a equalização de histograma e a limiarização. Outras operações também podem ser realizadas visando o realce das imagens através da alteração da escala de cinza.

2.3.3.1.2.1 Equalização do histograma

Equalização é uma maneira de manipulação de histograma que expande os níveis de cinza ao longo de todo intervalo. Consiste em uma transformação não linear que considera a distribuição acumulativa da imagem original, para gerar uma imagem resultante, cujo histograma será aproximadamente uniforme.

A equalização tem por objetivo reduzir a freqüência de ocorrência de valores de intensidade muito presentes e aumentar a freqüência de ocorrência dos valores relativamente raros. Desta forma o histograma da imagem terá uma aparência mais equilibrada, e possibilitará a obtenção de imagens com um número maior de detalhes visíveis, facilitando sua visualização.

A equalização do histograma pode ser conseguida particionando a *distribuição cumulativa de freqüências* (DCF = soma dos valores do histograma em cada nível de intensidade), em um maior número de intervalos para os níveis com maior freqüência e em menor número para os níveis mais raros.

A opção de equalização parte do princípio de que o contraste de uma imagem seria otimizado se todos os 256 possíveis níveis de intensidade fossem igualmente utilizados ou, em outras palavras, todas as barras verticais que compõem o histograma fossem da mesma altura. Obviamente isso não é possível devido à natureza das imagens. Contudo, uma aproximação é conseguida ao se espalhar os picos do histograma da imagem, deixando intocadas as partes mais “chatas” do mesmo. Esse processo é obtido através de uma função de transferência que tenha uma alta inclinação toda vez que o histograma original apresentar um pico, e uma baixa inclinação no restante do histograma.

De acordo com o que foi citado anteriormente, dado um histograma H(g) da imagem de entrada, com tons de cinza em $[g_0, g_k]$, o objetivo é encontrar uma função monotônica das intensidades dos pixels $q = T(g)$, tal que o histograma de saída G(q) é

uniforme em toda a escala de cinzas [q_0, q_k]. Sendo a imagem com NxM pixels, o histograma equalizado 'ideal' corresponde à distribuição uniforme f :

$$f = \frac{NxM}{q_k - q_0}$$

Assim, a equalização do histograma para imagens digitais é uma aproximação da transformação de intensidades dada por:

$$q = T(g) = \frac{q_k - q_0}{NxM} \sum_{i=g_0}^{g} H(i) + q_o$$

Um pseudocódigo, em linguagem C, explicando a implementação do algoritmo que calcula o histograma equalizado é mostrado a seguir.

```
char imagem[linhas][colunas];
int histograma[256];
int HC[256];
int linha, coluna, i;
// crie um vetor com 256 posições e inicialize com zeros
for (i = 0; i < 256; i++)
        histograma[ j ] = 0;
// calcule o histograma da imagem
for (linha = 0; linha < linhas; linha++)
    for(coluna = 0; coluna < colunas; coluna++)
            histograma [(int) imagem [linha][coluna]] ++;
// calcule o histograma cumulativo da imagem
Hc (0) = histograma (0);
Hc (p) = Hc(p-1) + histograma(p);  p = 1, 2, ..., 255
//construa a transformação
T(p) = roud  (g - 1)/NxM Hc(p)
//obtenho os tons de cinza da imagem
gq = T(gp);
```

Nas figuras a seguir será apresentado uma imagem com seu histograma e seu histograma equalizado.

FIGURA 2.12 - Imagem Original

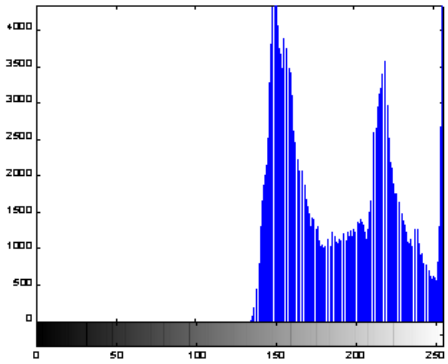


FIGURA 2.13 - Histograma original

FIGURA 2.14 - Imagem Equalizada

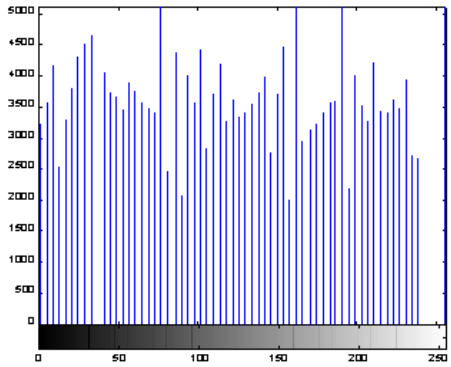


FIGURA 2.15 - Histograma Equalizado

2.3.3.1.2.2 Limiarização

A limiarização, ou *thresholding*, é uma das mais importantes abordagens para a segmentação de imagens [PAZ88], é uma técnica de análise por região particularmente útil para cenas que contêm objetos sobre um fundo contrastante. Sua implementação computacional geralmente é simples.

A limiarização de uma imagem, do tipo escala de cinza, consiste em se determinar um valor limiar (*threshold*) de forma que todos os valores inferiores a este limiar sejam levados para um mesmo valor base, usualmente 0. Todos os valores superiores a este limiar sejam levados para um mesmo valor de topo, usualmente 1.

Limiarização : $i(u, v) = 1$ se $i(u, v) > L$ (Limiar)

0 se $i(u, v) \leq L$

Desta forma é realizada a chamada *binarização da imagem*, onde, partindo de uma imagem do tipo escala de cinza, obtém-se uma imagem com apenas dois níveis de cinza.

A conversão de uma imagem com níveis de cinza para uma imagem com representação binária (dois tons) é importante para uma série de objetivos, tais como:

- identificar objetos e separá-los do fundo da imagem;
- analisar a forma da imagem quando é mais importante a forma que a intensidade dos pixels;

O diagrama da figura 2.16 representa um histograma típico de uma imagem de cor mais clara sob um fundo mais escuro. Ele é bi-modal, ou seja, pode ser representado como a combinação de dois histogramas característicos. Faz-se uma transformação da intensidade da imagem para que ela passe a ter só dois níveis distintos. Neste caso pode separar o objeto do fundo.

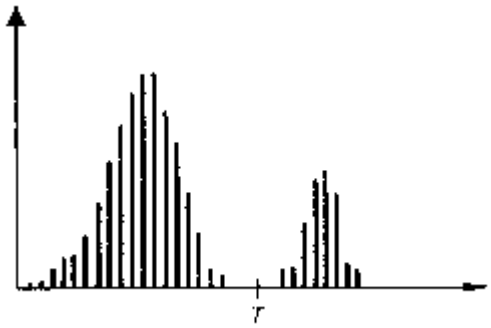


FIGURA 2.16 - Histograma e Threshold de uma Imagem

Esta transformação é chamada binarização, e pode ser descrita através da aplicação da função: s = T(r).

A função T(r) compara o sinal de entrada com um valor de limiar, escolhido como referência para a separação dos níveis de cinza. O sinal de saída, apresentado é obtido pela relação:

S = 1 para r > T

S = 0 para r < T

O histograma da imagem, após sua binarização, terá apenas dois tons com número de pixels diferentes de zero.

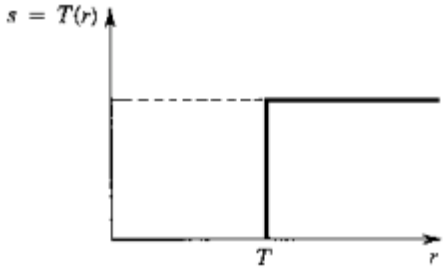


FIGURA 2.17 - Esquema para Binarização

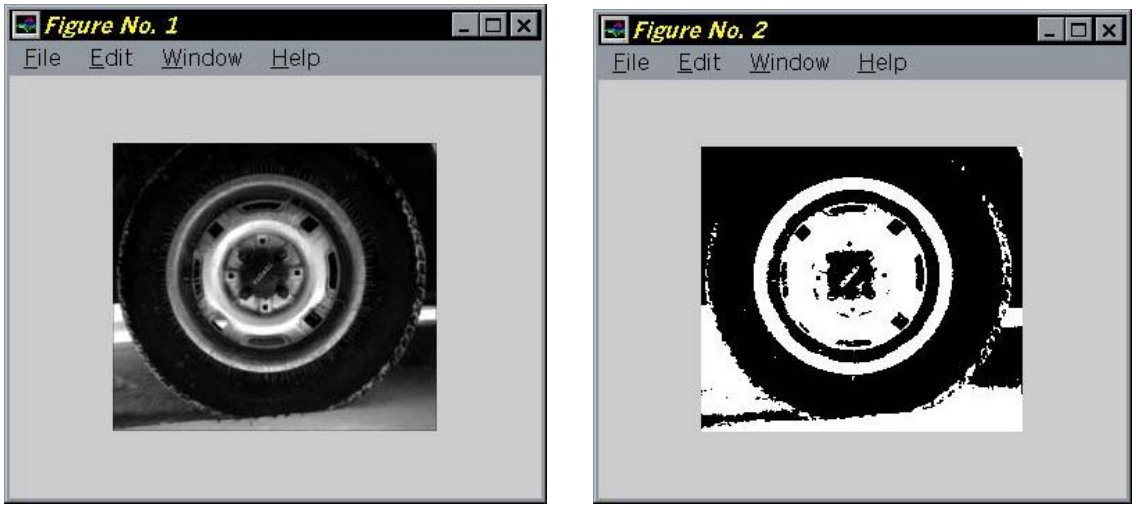


FIGURA 2.18 - Imagem Original e Imagem Binarizada

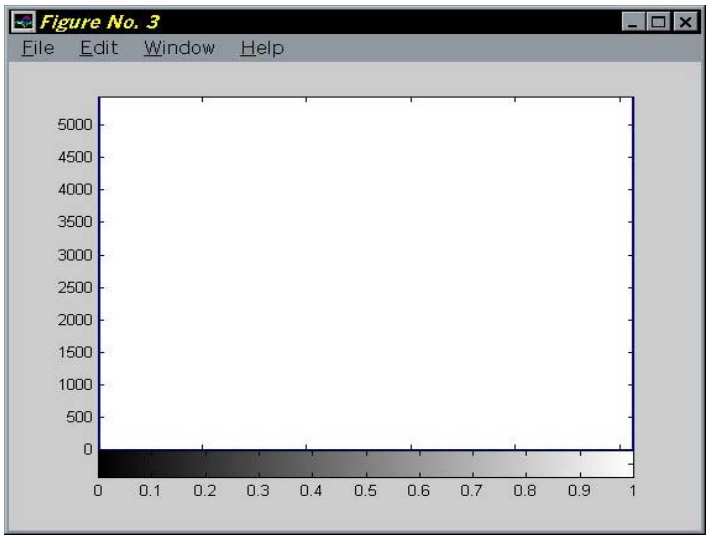


FIGURA 2.19 - Histograma da Imagem Binarizada

É evidente que a escolha adequada do valor de limiar é essencial para o bom funcionamento da técnica, e ainda, esta escolha é única para cada imagem.

Em muitos casos, principalmente quando não há um controle da iluminação sobre a imagem, o fundo não possui uma intensidade luminosa constante, e o contraste da imagem varia.

Neste caso, um valor de limiar que fornece um bom resultado em uma determinada região pode não ser adequado em outra. Como a iluminação sobre o objeto não é homogênea, um limiar global não funciona bem, causando uma perda de informação. É necessário um limiar variável que se adapte às diferentes condições de iluminação.

Como exemplo deste tipo de binarização tem-se a limiarização Niblack [NIB86]. Esta técnica é um algoritmo de binarização global adaptativa baseado em uma relação estatística entre cada pixel analisado e sua vizinhança.

O algoritmo é simples de ser implementado e calcula para cada *pixel* da imagem a média (2.1) e o desvio padrão (2.2) da vizinhança em torno dele, e a seguir, compara o valor do *pixel* analisado com o limiar *T(x,y)* (2.3). Se o valor do *pixel* for maior que *T(x,y)*, ele é considerado pertencente ao fundo, caso contrário, é considerado pertencente à algum objeto.

$$\mu(x,y) = \frac{1}{N.M}\sum_{i=0}^{N}\sum_{j=0}^{M} p(x,y) \qquad (2.1)$$

$$\sigma(x,y) = \sqrt{\frac{1}{N.M}\sum_{i=0}^{N}\sum_{j=0}^{M}\left(p(x,y)-\mu(x,y)\right)^2} \qquad (2.2)$$

$$T(x,y) = -\alpha.\sigma(x,y) + \mu(x,y) \qquad (2.3)$$

Para a definição do tamanho da janela é preciso levar em conta a preservação de detalhes locais e a supressão de ruídos indesejáveis. Nos testes feitos por Trier [TRI95], bem como na literatura pesquisada, é padrão a utilização de uma janela quadrada de dimensões 15x15 *pixels* e uma valor de α constante igual a 0.2.

As imagens apresentadas nas figuras 2.20 e 2.21 mostram o resultado da aplicação da técnica de limiarização Niblack sobre as imagens.

FIGURA 2.20 - Imagem Original FIGURA 2.21 - Imagem Binarizada

De acordo com [SOU2000] duas conclusões importantes devem ser observadas. A primeira, é que à medida que a janela de amostragem cresce o ruído vai sendo suprimido. Isto ocorre porque a binarização Niblack opera em cima da distribuição estatística dos *pixels* e suas vizinhanças, portanto, a amostra analisada deve possuir dimensões próximas das do objeto que se procura segmentar. A segunda conclusão é que as formas originais do objeto permanecem inalteradas para qualquer uma das

dimensões utilizadas. Isto ocorre porque nas bordas do objeto a janela de amostragem tem, em média, metade de seus *pixels* preenchidos pelos *pixels* do objeto e a outra metade pelos *pixels* do fundo.

2.3.3.2 Filtragem Espacial

As imagens digitais podem sofrer uma série de operações denominadas operações com filtros. Estas operações realizam tarefas como: ressaltar elementos, suavizar ou aumentar o contraste, detectar bordas, remover o ruído, entre outras.

Um filtro é uma rede que transforma um sinal de entrada em um determinado sinal de saída desejado, como na figura 2.22. Os sinais podem ser considerados em um domínio de tempo ou em um domínio de freqüência, da mesma forma, os requisitos de saída do filtro podem ser gerados em termos de tempo ou freqüência. A relação entre filtragem no domínio espacial e no domínio freqüência esta representada na figura 2.23.

FIGURA 2.22 - Representação Simbólica de um Filtro

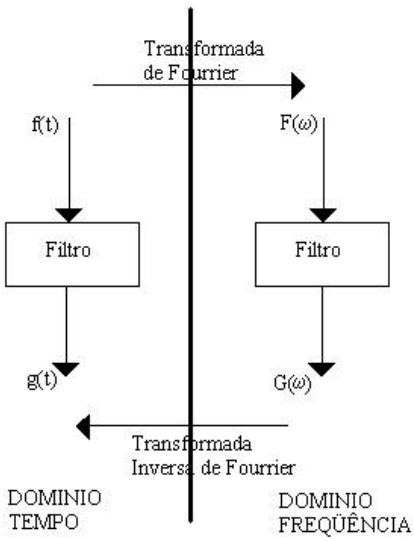


FIGURA 2.23 - Relação entre filtro no domínio tempo e no domínio freqüência

O termo domínio espacial refere-se ao agregado de pixels que compõem uma imagem, e métodos no domínio espacial são métodos que operam diretamente sobre estes pixels. O uso de máscaras espaciais para processamento de imagens é usualmente chamado *filtragem espacial*.

As operações no domínio espacial são realizadas diretamente com os pixels da imagem, o que é uma vantagem, pois a imagem não sofre transformações prévias e posteriores para poder ser processada, ao contrário do que ocorre com as operações

realizadas no domínio freqüência onde a imagem deve ser transformada do domínio espacial para o domínio freqüência para poder ser tratada e, então, transformada novamente para o domínio espacial.

2.3.3.3 Filtragem no Domínio Freqüência

No realce no domínio freqüência deve-se computar a transformada de Fourier da imagem a ser realçada, multiplicar o resultado por uma função filtro de transferência, e então tomar a transformada inversa para produzir a imagem realçada.

As idéias de borramento, através da redução do conteúdo de alta freqüência ou do aguçamento através do aumento da magnitude dos componentes de alta freqüência relativamente aos componentes de baixa freqüência, originam-se dos conceitos diretamente relacionados à transformada de Fourier. Na prática, pequenas máscaras espaciais são mais freqüentemente usadas do que a transformada de Fourier, devido a sua simplicidade de implementação e velocidade [GON93]. Entretanto, uma compreensão dos conceitos do domínio da freqüência é essencial para a solução de problemas que não são facilmente tratáveis por técnicas espaciais. Pode-se citar a abordagem por filtragem homomórfica aplicada neste trabalho.

2.3.3.3.1 Filtragem Homormófica

O modelo de iluminação - reflectância, descrito na seção 2.2.1, é utilizado como base para este método, no domínio da freqüência, que busca melhorar a aparência da imagem através da compressão do intervalo de brilho e realce de contraste simultâneos. A aproximação de problemas de filtragem não linear através do princípio da superposição generalizada tem apresentado sucesso em algumas aplicações práticas.

Uma imagem *f(x,y)* pode ser expressa em termos dos seus componentes de iluminação e reflectância por meio da relação:

$$f(x,y) = i(x,y)\ r(x,y) \qquad (2.4)$$

A equação (2.4) não pode ser utilizada diretamente para que possa operar separadamente sobre os componentes de freqüência da iluminação e reflectância, porque a transformada de Fourier do produto de suas funções não é separável [GON93], ou seja:

$$\Im\{f(x,y)\} \neq \Im\{i(x,y)\}\ \Im\{r(x,y)\}$$

Assim, tem-se: $z(x,y) = \ln f(x,y)$

$$= \ln i(x,y) + \ln r(x,y)$$

Então, $\Im\{z(x,y)\} = \Im\{\ln f(x,y)\}$

$$= \Im\{\ln i(x,y)\} + \Im\{\ln r(x,y)\}$$
$$= Z(u,v) = I(u,v) + R(u,v)$$

em que I(u,v) e R(u,v) são as transformadas de Fourier de ln $i(x,y)\}$ e ln $r(x,y)$, respectivamente. Se processarmos Z(u,v) através de uma função de filtro H(u,v) tem-se:

$$S(u,v) = H(u,v)\, Z(u,v)$$
$$= H(u,v)I(u,v) + H(u,v)R(u,v)$$

em que S(u,v) é a Fourier do resultado.

No domínio espacial, $s(x,y) = \Im^{-1}\{S(u,v)\}$

$$= \Im^{-1}\{H(u,v)I(u,v)\} + \Im^{-1}\{H(u,v)\, R(u,v)\}$$

Sejam:

$$i'(x,y) = \Im^{-1}\{H(u,v)I(u,v)\}$$
$$r'(x,y) = \Im^{-1}\{H(u,v)\, R(u,v)\}$$

Tem-se:

$$s(x,y) = i'(x,y) + r'(x,y)$$

Finalmente, como z(x,y) foi calculado através do logaritmo da imagem original f(x,y), a operação inversa produz a imagem realçada desejada g(x,y), isto é:

$$g(x,y) = \exp[s(x,y)]$$
$$= \exp[i'(x,y)] \,.\, \exp[r'(x,y)]$$
$$= i_o(x,y) \,.\, r_o(x,y)$$

onde i_o e r_o são as componentes de iluminação e reflectância da imagem de saída.

2.3.3.3.2 Filtragem Butterworth

A função de transferência do filtro de Butterworth de ordem *n* e com freqüência de corte posicionada a uma distância D_0 da origem é definida pela relação (2.5).

$$H_{(u,v)} = \frac{1}{1 + 0{,}414\left[D(u,v)/Do\right]^{2n}} \qquad (2.5)$$

Em que D(u,v) é definido por:

$$D(u,v) = (u^2 + v^2)^{1/2}$$

Ao contrário do filtro passa baixa ideal, a função de transferência do FPBB não possui uma descontinuidade abrupta que estabeleça um corte claro entre as freqüências passadas e filtradas.

3 Reconhecimento de Padrões

3.1 Introdução

O objetivo do reconhecimento de padrões junto ao processamento de imagens é extrair, detectar e identificar elementos em uma cena. Desta forma, procura-se imitar o ser humano e suas habilidades, criando dispositivos e algoritmos capazes de realizar as mesmas funções realizadas pelo homem como localizar objetos, classificar padrões e detectar relações entre estes.

3.2 Redes Neurais

Neste capítulo é feita uma revisão sobre redes neurais artificiais. Inicialmente tem-se uma introdução e um breve histórico das redes neurais. A seguir são descritos os componentes básicos destes modelos e como estes se relacionam. São apresentadas então as principais arquiteturas das redes e o conceito de aprendizado em redes neurais. Por fim é apresentada a rede Feedfoward Multicamada de Perceptrons com o algoritmo de treinamento Backpropagation além dos métodos de implementação.

3.2.1 Introdução

O final da década de 80 marcou o ressurgimento da área de Redes Neurais Artificiais (RNAs). Esta forma de computação não algorítmica é caracterizada por sistemas que, em algum nível, lembram a estrutura do cérebro humano. Por não ser baseada em regras ou programas, a computação neural se constitui em uma alternativa à computação algorítmica convencional[BRA97].

RNAs são sistemas paralelos distribuídos compostos por unidades de processamento simples que computam determinadas funções matemáticas. Tais unidades são dispostas em uma ou mais camadas e interligadas por um grande número de conexões, geralmente unidirecionais. Na maioria dos modelos estas conexões estão associadas a pesos, os quais armazenam o conhecimento representado no modelo e servem para ponderar a entrada recebida por cada neurônio da rede.

Em RNAs, o procedimento usual na solução de problemas passa inicialmente por uma fase de aprendizagem, onde um conjunto de exemplos é apresentado para a rede, a qual extrai as características necessárias para representar a informação fornecida. Essas características são utilizadas posteriormente para gerar respostas para o problema.

A capacidade de aprender através de exemplos e de generalizar a informação aprendida são, sem dúvida, os atrativos principais da solução de problemas através de

RNAs. A generalização, que está associada à capacidade da rede aprender através de um conjunto reduzido de exemplos e posteriormente fornecer respostas coerentes para dados não conhecidos, é uma demonstração de que a capacidade das RNAs vai muito além do que simplesmente mapear relações de entrada e saída. Outras características importantes são a capacidade de auto-organização e de processamento temporal que, aliadas a capacidade de atuar como mapeadores universais, fazem das RNAs uma ferramenta computacional extremamente poderosa e atrativa para a solução de problemas complexos.

Alguns estudos da neurofisiologia consideram que a riqueza computacional do cérebro humano vem do grande número de neurônios que estão interconectados por uma rede complexa de sinapses [CAR88].

A velocidade de processamento destes componentes individuais é baixa se comparada com a velocidade dos componentes digitais dos computadores tradicionais. Tipicamente neurônios são cinco a seis vezes mais lentos do que as portas lógicas de silício; eventos em um chip de silício ocorrem em nanosegundos (10e-9), enquanto que em um neurônio ocorrem em milisegundos (10e-3). A diferença, entretanto, é largamente superada pela imensa quantidade de neurônios existentes operando em paralelo [SIM90]. Estima-se que existam cerca de 10^{11} a 10^{14} neurônios operando em paralelo no cérebro humano. Cada um destes está conectado através de 10^{3} a 10^{4} sinapses em média [COT85].

Tais características permitem ao cérebro humano executar rapidamente certas funções (por exemplo, reconhecer fisionomias) que os computadores convencionais não conseguem realizar com o mesmo desempenho. Na tabela 1, apresentada em [MOL98], tem-se uma comparação entre os computadores tradicionais e o cérebro humano. Esta comparação nos permite ter uma idéia mais clara sobre a capacidade adaptativa do cérebro humano, em contraste com a rigidez e a precisão dos computadores convencionais.

TABELA 3.1 - Diferenças Entre o Computador e o Cérebro Humano [Molz98]

Características	*Computador*	*Cérebro Humano*
Elementos Computacionais	Processadores	neurônio simples
Velocidade de Processamento	10^{-9} segundos	10^{-3} segundos
Tipo de Processamento	serial	paralelo
Confiabilidade dos Elementos	Confiável	não-confiável
Tolerância a Falhas	Quase nenhuma	grande

Tipo de Sinal	Precisos, simbólicos	imprecisos
Tipo de Controle	Centralizado	distribuído
Armazenamento de Informações	rígido	adaptativo

3.2.2 Histórico

A área de redes neurais é relativamente nova. Em 1943, McCulloch e Pitts [McC43] criaram o primeiro modelo computacional, o Psychon, que é um sistema lógico de dois estados que gera um sinal binário de saída quando as entradas somadas ultrapassam um valor limite de excitação. Entretanto, este modelo não previa a capacidade de adaptação.

Em 1949, D. O. Hebb publicou sua importante obra *'The organization of Behavior'* [HEB49]. Este trabalho não resultou diretamente em um modelo específico e bem formalizado de uma rede neural, mas sim em uma análise das características e formas de comportamento que uma rede neural deveria possuir, ele estabeleceu que as sinapses mais freqüentemente ativadas devem ter maior chance de se tornarem ativas novamente. Em 1959, Frank Rosenblatt criou o Perceptron que tem até hoje uma grande influência nos estudos sobre RNAs [ROS59]. Nesta época, também foram desenvolvidos outros modelos similares ao Perceptron como é o caso do Adaline, criado por Bernard Widrow em 1962 [WID62]. Estes modelos são baseados na correção de erros e formam uma importante classe de RNAs.

Entre 1969 e o inicio da década de 80 as redes neurais perderam o entusiasmo. A publicação feita por Misnky e Paper provou matematicamente que os modelos de RNAs usados até então não eram capazes de aprender uma simples função lógica, "XOR" [MIN69].

O ressurgimento do interesse pela área veio em 1982 com o modelo de Hopfield o qual utilizava os conceitos de aprendizado definidos por Hebb e que chamou atenção das propriedades associativas das RNAs [HOP82]. O modelo multinível, que utiliza a regra de aprendizado *backpropagation* foi apresentado por Parker [PAR85] e Rumelhart e McClelland [McC86]. Vários outros modelos surgiram e houve então uma nova explosão de interesse pelas RNAs na comunidade internacional. Outro fator responsável pela retomada de interesse na área foi o avanço da microeletrônica, que vem permitindo a realização física de modelos de neurônios e sua interconexão de modo antes impensável.

3.2.3 Modelo de um Neurônio

Assim como o sistema nervoso é composto por bilhões de células nervosas, a rede neural artificial também é formada por unidades que nada mais são do que pequenos módulos que simulam o funcionamento dos neurônios. Estes módulos devem funcionar de acordo com os elementos em que foram inspirados, recebendo e transmitindo informações.

3.2.3.1 Neurônios Biológicos

Os neurônios são divididos em três seções: *o corpo da célula, os dentritos e o axônio*, cada um com funções específicas, porém complementares. O corpo do neurônio mede apenas alguns milésimos de milímetros e os dentritos apresentam poucos milímetros de comprimento. O axônio, contudo, pode ser mais longo e, em geral, apresentam calibre uniforme [BRA97].

Os dentritos têm por função receber as informações, ou impulsos nervosos, oriundas de outros neurônios, e conduzi-las até o corpo celular. Ali as informações são processadas e novos impulsos são gerados. Estes impulsos são transmitidos a outros neurônios, passando através do axônio até os dentritos do neurônio seguinte. O ponto de contato entre a terminação axônica de um neurônio e os dentritos do outro é chamado de *sinapse*. É pelas *sinapses* que os neurônios se unem funcionalmente formando redes neurais. O efeito das sinapses é variável, e é esta variação que dá ao neurônio a capacidade de adaptação. A figura 3.1 ilustra, de forma simplificada, os componentes do neurônio.

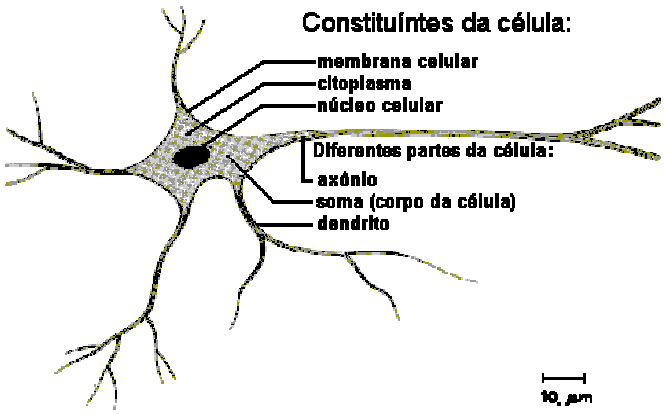


FIGURA 3.1 - Componentes do Neurônio Biológico

3.2.3.2 Neurônios Artificiais: Modelo MCP

O modelo de neurônio proposto por McCulloch e Pitts [McC43] é uma simplificação do que se sabia a respeito do neurônio biológico naquela época. A sua

descrição matemática resultou em um modelo com *n* terminais de entrada $x_1, x_2, ..., x_n$ (que representam os dentritos), e apenas um terminal de saída *y* (representando o axônio). Para emular o comportamento das sinapses, os terminais de entrada do neurônio tem pesos (resistores) acoplados $w_1, w_2, ..., w_n$, cujos valores podem ser positivos ou negativos, dependendo das sinapses correspondentes serem inibitórias ou excitatórias. O efeito de uma sinapse particular *i* no neurônio pós-sináptico é dado por: $x_i w_i$. Uma descrição do modelo está ilustrado na figura 3.2.

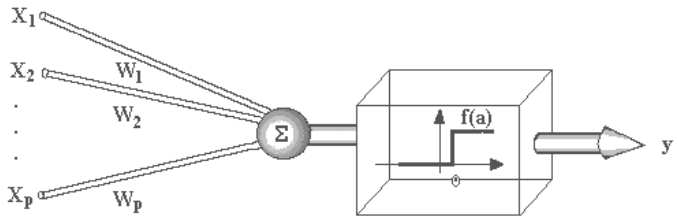


FIGURA 3.2 - Neurônio de McCulloch e Pitt

Um neurônio biológico dispara quando a soma dos impulsos que ele recebe ultrapassa o seu limiar de excitação (*threshold*). O corpo do neurônio, por sua vez, é emulado por um mecanismo simples que faz a soma dos valores $x_i w_i$ recebidos pelo neurônio (soma ponderada), e decide se o neurônio deve ou não disparar (saída igual a 1 ou a 0) comparando a soma obtida ao limiar ou *threshold* do neurônio. No modelo MCP, a ativação do neurônio é obtida através da aplicação de uma função de ativação, que ativa a saída ou não dependendo do valor da soma ponderada das suas entradas. Na descrição original do modelo MCP, a função de ativação é dada pela função de limiar descrita na equação 1, e o neurônio terá sua saída ativada quando:

$$\sum_{i=1}^{n} x_i w_i \geq \theta \qquad (1)$$

onde *n* é o número de entradas do neurônio, w_i é o peso associado à entrada i, e θ é o limiar do neurônio.

3.2.4 Componentes de uma Rede Neural Artificial

De acordo com Rumelhart [RUM86], uma RNA pode ser descrita por oito elementos principais:

- Um conjunto de unidades de processamento;
- Um estado de ativação;
- Uma função de saída;
- Um padrão de interconexão;

- Uma regra de propagação;
- Uma regra de ativação;
- Uma regra de aprendizado;
- Um ambiente onde o sistema deve funcionar;

3.2.4.1 Unidades de Processamento

Os neurônios constituem o meio de representação do conhecimento existentes nas RNAs. Os nós podem representar pontos (*pixels*), caracteres (*letras, números*), palavras ou outros conceitos, dependendo da aplicação. Na figura 3.3 tem-se uma ilustração de um neurônio como unidade de limiar. As entradas que chegam a ele representam os dentritos. Cada dentrito recebe um sinal que é ponderado e, depois somado com os outros sinais dos demais dentritos, por fim é processado através da função limiar, f(), a qual produz um sinal de saída. As unidades de processamento da rede serão designadas pela letra **u**, seguida de um índice **i** que indica a posição que o neurônio ocupa na rede.

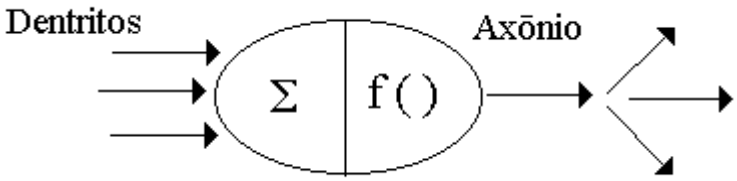


FIGURA 3.3 - Neurônio como unidade limiar

3.2.4.2 Estado de Ativação

Cada célula $\mathbf{u_i}$ da rede computa um estado de ativação, que é um valor numérico líquido de saída. O cálculo desta ativação é computado a partir das ativações das células conectadas diretamente a este nó, e dos correspondentes pesos destas conexões e função de ativação.

O estado de ativação de todas as unidades da rede, ou seja, o estado de ativação do sistema, especifica o que está sendo representado nas redes em um determinado instante **t** qualquer. Este estado de ativação do sistema pode ser representado por um vetor **a(t)**.

3.2.4.3 Função de Saída

As unidades interagem entre si através de um valor que é transmitido pelas sinapses. Este valor é determinado pela ativação da unidade estimuladora. Formalmente o valor de saída é dado por uma função do tipo $\mathbf{o_i(t) = g(a_i(t))}$.

3.2.4.4 Padrão de Interconexão

Pode-se representar o padrão de interconexão da rede por uma matriz de pesos **w**, onde o elemento $\mathbf{w_{ij}}$ corresponde à influência da célula $\mathbf{u_j}$ sobre a célula $\mathbf{u_i}$. Conexões, também chamadas sinapses, com pesos positivos, indicam o reforço na ativação do neurônio $\mathbf{u_i}$. Estas conexões são chamadas excitatórias. Sinapses com pesos negativos, chamadas de inibitórias, indicam inibição na ativação da célula $\mathbf{u_i}$. O conjunto das ligações excitatórias e inibitórias existentes na rede determina o comportamento da mesma.

3.2.4.5 Regra de Propagação

Cada célula $\mathbf{u_i}$ computa sua nova ativação através de uma regra de propagação. Em geral, ela é definida como sendo uma função da soma dos produtos das entradas pelos pesos das células $\mathbf{u_j}$ que estão diretamente conectadas à célula $\mathbf{u_i}$ conforma a equação 2.

$$fi = F\left(\sum_{j=1}^{n} w_{ij} * u_j - \theta_i\right) \qquad (2)$$

Onde: $\mathbf{u_j}$ é o estado da j-ésima unidade

$\mathbf{w_{ij}}$ é o peso da conexão da j-ésima unidade para a i-ésima unidade

$\theta_\mathbf{i}$ é o limiar da i-ésima unidade.

3.2.4.6 Função de Ativação

O sinal de saída de um neurônio é calculado a partir da sua ativação, pela função de ativação. Na figura 3.3 o bloco **f()** representa a função de ativação. Na prática, as funções de ativação mais utilizadas são as sigmoidais diferenciáveis, representadas pela função logística ([0;1]) e pela tangente hiperbólica ([-1; 1]). As funções logística e hiperbólica permitem uma melhor convergência dos valores de saída durante o treinamento, fornecendo melhores resultados durante a operação das redes. As aproximações por função linear (f(x)=x) e função de Siebert [KOV96] também podem ser aplicadas.

3.2.4.7 Regra de Aprendizado

Redes Neurais Artificiais possuem capacidade de aprender por exemplos e fazer interpolações e extrapolações do que aprenderam. No aprendizado conexionista não se procura obter regras como na abordagem simbólica da Inteligência Artificial, mas determinar a intensidade de conexões entre neurônios. Um conjunto de procedimentos

bem definidos para adaptar os parâmetros de uma RNA para que a mesma possa aprender uma determinada função é chamado *algoritmo de aprendizado*.

As regras de aprendizado determinam como os pesos das sinapses das redes são alterados através da experiência. Em geral, as regras de aprendizado podem ser consideradas como uma variante da regra de Hebb [KOS92] em que a alteração da eficiência sináptica é a base do aprendizado.

3.2.4.8 Ambiente

O último componente de RNAs é o ambiente onde a rede deve funcionar. É necessário especificar a natureza do ambiente, estabelecendo seus possíveis padrões de entrada e saída.

3.2.5 Principais Arquiteturas das RNAs

A definição da arquitetura de uma RNA é um aspecto importante na sua concepção, uma vez que ela restringe o tipo de problema que pode ser tratado pela rede [BRA97]. Fazem parte da definição da arquitetura os seguintes parâmetros: número de camadas da rede, número de neurônios em cada camada, tipo de conexão entre os neurônios e topologia da rede.

Quanto ao número de camadas, pode-se ter redes de uma ou múltiplas camadas. Quanto ao tipo de conexões existentes entre os neurônios pode-se ter rede acíclica (*feedforward*) ou cíclica (*feedback*).

1- Rede de Camada Única:

Redes com uma única camada de neurônios só conseguem resolver problemas linearmente separáveis. São redes onde só existe um nó entre qualquer entrada e qualquer saída da rede, a figura 3.4 ilustra este tipo de rede.

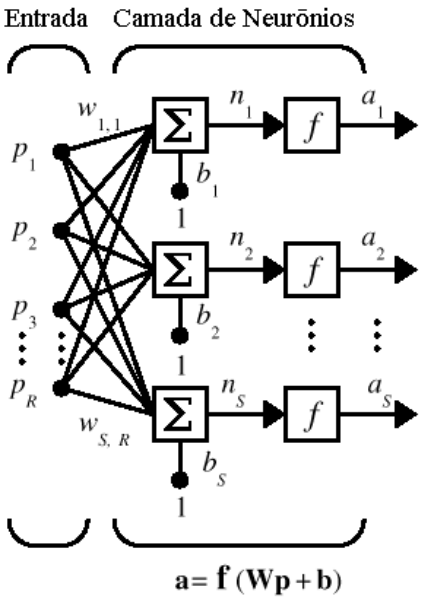


FIGURA 3.4 - Rede Neural Artificial com uma Camada de Neurônios

Nesta rede, os elementos do vetor de entrada *p* se conectam aos neurônios existentes na rede através de uma matriz de pesos *w* , e tem-se, por fim, um vetor coluna *a* representando a saída da rede.

2- Rede de Múltiplas Camadas:

Uma rede neural pode ter várias camadas. Cada camada possui uma matriz de pesos, um vetor **b** de limiares e um vetor de saída, **a**.

Nas RNAs com múltiplas camadas existe mais de um neurônio entre alguma entrada e alguma saída da rede. A figura 3.5 exemplifica esta arquitetura.

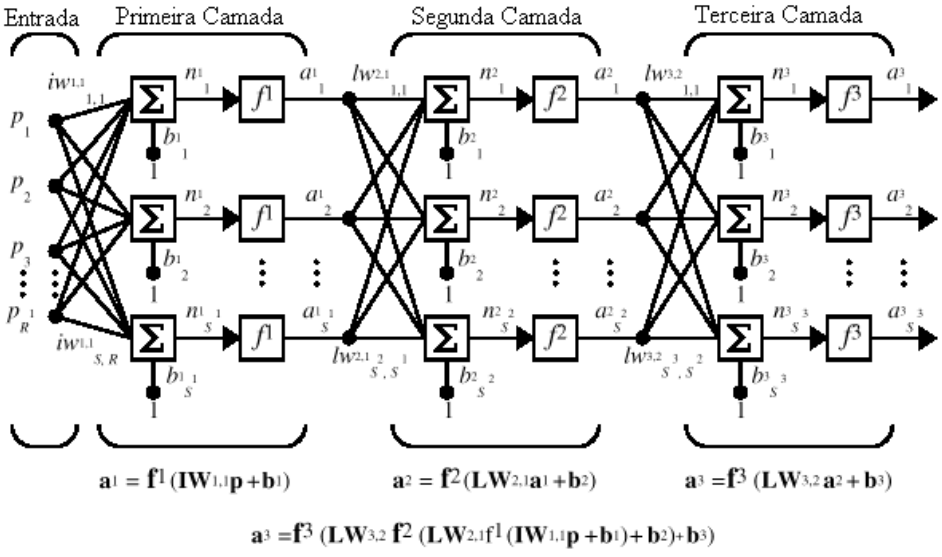


FIGURA 3.5 - Rede Neural Artificial com múltiplas camadas de neurônios

3- Rede *Feedforward* ou acíclica:

Redes f*eedforward* são redes neurais onde a saída de um neurônio na *i*-ésima camada da rede não pode ser usada como entrada de neurônios em camadas de índice menor ou igual a *i*. A figura 3.6 apresenta esta tipo de rede.

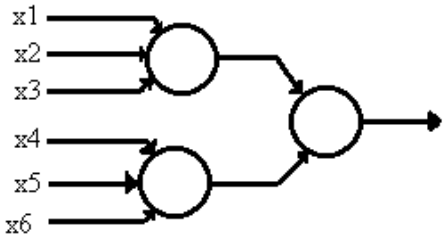


FIGURA 3.6 - Rede *Feedforward*

4- Rede *Feedback* ou cíclica:

Redes *feedback* são redes onde a saída de um neurônio na *i*-ésima camada da rede é usada como entrada de neurônios em camadas de índice menor ou igual a *i*; este tipo de rede pode ser observado na figura 3.7.

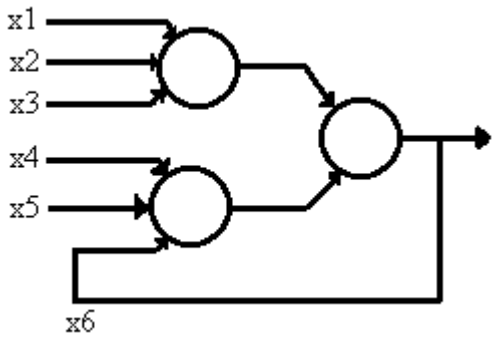


FIGURA 3.7 - Rede *Feedback*

Para se obter uma boa generalização com RNAs, deve-se fornecer para a rede a maior quantidade possível de informação a respeito do problema a ser solucionado. Isto significa que uma grande quantidade de neurônios será usada para sub-tarefas específicas. Contudo, por problemas de complexidade computacional (tempo e espaço), deve-se buscar reduzir ao mínimo o número de neurônios e a quantidade de conexões entre eles. Portanto é importante a definição de algoritmos que não somente otimizem os pesos para uma dada arquitetura, como também, otimizem a própria arquitetura.

Uma outra maneira de classificar os modelos de redes neurais é de acordo com o tipo de treinamento e regra de aprendizagem. O treinamento pode ser supervisionado ou não supervisionado. O treinamento supervisionado consiste em apresentar à rede um padrão a ser reconhecido juntamente com a resposta que a rede deve fornecer ao

reconhecer este determinado padrão. Geralmente, neste tipo de treinamento temos uma regra de aprendizado do tipo correção de erros. Esta regra está baseada no princípio de adaptação e correção dos pesos de atuação de cada neurônio, até que este responda da maneira desejada.

O treinamento não supervisonado, ou auto-aprendizado, consiste apenas em apresentar os padrões que se quer reconhecer à rede e esta deverá ser capaz de agrupar os padrões que possuem propriedades similares.

3.2.6 Técnicas de Aprendizado

Dentre as diversas técnicas de aprendizado utilizadas em RNAs tem-se a técnica de aprendizado por correção de erro. Nesta técnica os pesos das conexões entre os nós são ajustados de acordo com a diferença entre os valores desejados e computados de cada nó da camada de saída. Outra técnica existente é a técnica de aprendizado por reforço. Assim como a anterior esta é uma técnica de aprendizado supervisionado. Na técnica de aprendizado por reforço os pesos são recompensados quando o sistema executa ações apropriadas e punidos caso ele não os execute.

Tem-se ainda a técnica de aprendizado estocástica que utiliza processos aleatórios, probabilidade e relações de energia para ajustar os pesos dos arcos. A técnica de aprendizado denominada regra de Hebb onde os ajustes dos pesos das conexões é realizado em função da relação de valores dos dois nós que ele conecta. Pode ser aplicado tanto no aprendizado supervisionado quanto ao aprendizado não-supervisionado.

Outra técnica de aprendizado não supervisionada é a técnica de processos competitivos e cooperativos. É uma técnica onde os processos competitivo e cooperativo são descritos em termos de redes com conexões recorrente auto-excitáveis. Estes arcos podem ser inibitórios dos nós vizinhos, competitivo, e ou excitatórios dos vizinhos, cooperativo. Tem-se também a técnica dos sistemas conectados aleatoriamente (SCA) que, assim como a anterior, é uma técnica de aprendizado não supervisionado. Esta técnica é utilizada para suportar a teoria de que a mente é uma rede conectada aleatoriamente quando vista do nível macroscópico.

3.2.7 Redes Multicamadas e o Algoritmo Backpropagation

A rede neural MLP (*Multi-Layer Perceptron*) merece ser vista com detalhamento maior visto que é o modelo utilizado na implementação do sistema desenvolvido neste trabalho. Este modelo tem sido aplicado com sucesso para a solução de vários problemas utilizando o popular algoritmo *Backpropagation*. Basicamente, o

processo de retropropagação do erro é constituído de dois passos conforme ilustra a figura 3.9. No primeiro passo, *forward*, o vetor de entrada é aplicado aos nós de entrada da RNA e este é propagado camada por camada até a camada de saída da RNA. Durante este processo não ocorre nenhum ajuste nos pesos da conexões. Durante o segundo passo, *Backward*, os pesos são ajustados de acordo com a regra de conexão de erros. O resultado atual calculado pela RNA é subtraído do desejado produzindo o erro. Este sinal de erro é propagado da última para a primeira camada sendo utilizado para o ajuste dos pesos em cada camada. Este ajuste de pesos é feito de forma a minimizar o erro.

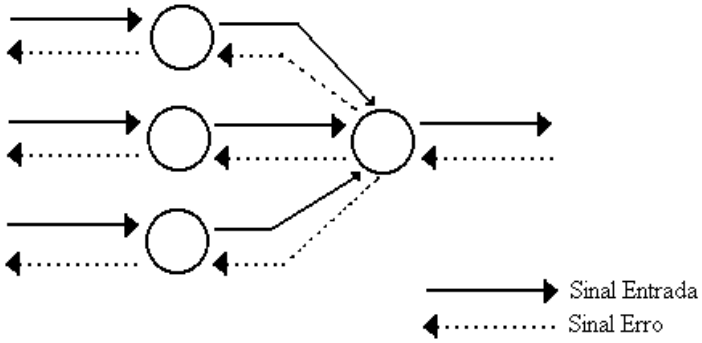


FIGURA 3.8 - Fluxos dos Sinais de entrada e de erro

Uma rede Perceptron de múltiplas camadas tem três características importantes:

- O modelo de cada neurônio da RNA inclui, normalmente, uma não-linearidade em sua saída. Tipicamente é utilizada uma função não-linear sigmoidal.
- A RNA pode ser constituída de uma ou mais camadas escondidas dando a esta uma maior capacidade de mapear funções com maior nível de complexidade.
- A RNA possui um alto grau de conectividade.

Este modelo, também conhecido como BPN (*backpropagation network*), é uma rede *feedforward*, com treinamento supervisionado, formada por três ou mais níveis: um nível de entrada (*input layer*), um ou mais níveis intermediários ou ocultos (*hidden layers*) e um nível de saída (*output layer*), conforma a figura 3.9.

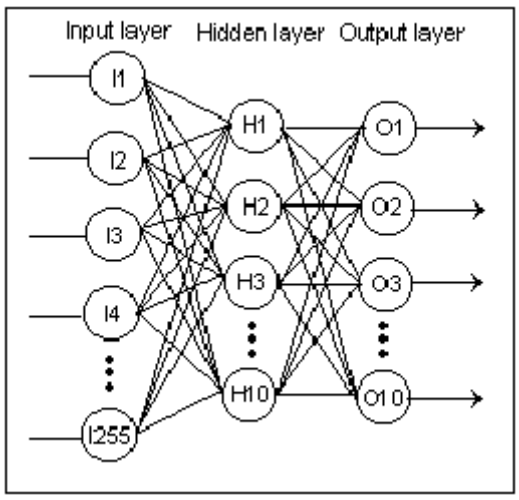


FIGURA 3.9 - Rede Multicamadas Feedforward com treinamento Backpropagation

Na grande maioria dos caos práticos, a camada oculta tem um só nível. O número de neurônios do nível de entrada e saída pode ser estimado de acordo com as características do problema. Entretanto, não existe uma técnica precisa para estimar o número de neurônios da camada oculta, devendo ser estabelecido de uma maneira intuitiva [LIB97] .

3.2.8 Modelos de Implementação

Basicamente há dois modos de implementação de redes neurais artificiais: por *software* e por *hardware*.

O primeiro deles é obtido através da simulação, por meio de uma linguagem de programação, em um *hardware* seqüencial, geralmente um computador de uso genérico. Através deste modo, consegue-se de uma forma mais ágil a implementação da rede neural para uma determinada aplicação. Contudo, devido à natureza serial da execução das instruções do programa, o desempenho da rede é mais lento que o segundo modo de implementação.

Como o tempo de desenvolvimento de um protótipo em software é mais curto que o tempo de desenvolvimento de um protótipo em hardware, este modo é ideal para a realização de simulações de redes neurais discretas ou contínuas, estas últimas se dando através de métodos computacionais para a resolução de equações diferenciais que geralmente norteiam os modelos de redes neurais contínuas [MOL98].

A implementação em *software* garante ainda, uma precisão maior que a implementação em hardware, conforme [KOS92].

Na implementação em hardware, o tempo de desenvolvimento de um protótipo é maior, mas devido a sua alta taxa de processamento paralelo torna-se ideal para aplicações que envolvam o processamento de sinais em tempo real.

4 Sistema de Identificação Automática de Veículos

4.1 Introdução

A área de visão computacional (VC) tem obtido grandes avanços em pesquisa e desenvolvimento de aplicações nos últimos anos. Além do surgimento de novos algoritmos e técnicas para processamento digital de imagens e reconhecimento de padrões, avanços em tecnologias de computadores como memórias de baixo custo, computadores mais velozes e processamento paralelo estão tornando possível a utilização de sistemas complexos de visão artificial em tempo real.

A identificação automática de veículos é uma importante aplicação de sistemas de visão artificial. O sucesso no desenvolvimento de pesquisa nesta área envolve processamento de sinais e técnicas de inteligência artificial.

O constante crescimento do volume de tráfego de automóveis e a limitada capacidade dos sensores convencionais têm levado os especialistas desta área a recorrer a técnicas de identificação automática de veículos para obter dados relativos ao escoamento de tráfego.

Dentre as diversas aplicações no trânsito, a identificação de veículos através da leitura de sua placa de licença vem conquistando cada vez mais espaço. No início dos anos cinqüenta, este conceito era usado para estudar o tempo de duração de viagens entre origem e destino. Os primeiros métodos utilizados eram baseados em observadores que anotavam as placas dos veículos e os tempos correspondentes em um papel. As placas eram manualmente comparadas mais tarde, e os tempos de viagem calculados [TUR51]. O crescente avanço tecnológico tem aumentado substancialmente a precisão e facilidade desta técnica.

A violência no trânsito nas cidades de médio e grande porte cada vez mais se torna um problema crítico. Diversos são os fatores que contribuem para isto, sendo o principal deles a velocidade excessiva dos automóveis. Atualmente existem 16 "*pardais*" nas ruas de Porto Alegre que captam imagens de automóveis infratores, ou seja, carros que ultrapassaram a velocidade permitida em um determinado local. Estas imagens são posteriormente analisadas de forma que a placa seja encontrada e a multa emitida. Esta análise ocorre de forma manual, o que ocasiona uma demora no processamento, tendo em vista a grande quantidade de dados a ser analisada, o que consequentemente retarda o envio da multa. Pelo código de transito brasileiro a multa

deve ser emitida até um mês após a ocorrência desta. Os fatos citados acima evidenciam a importância do desenvolvimento de um sistema capaz de processar as imagens capturadas pelos "*pardais*". Este sistema deve ser capaz de, a partir da imagem digitalizada, verificar a existência da placa de licença e localizá-la na imagem, extrair os caracteres presentes na placa e reconhecê-los melhorando a taxa de acerto e o tempo de processamento em relação a implementação SIAV1.0.

Um sistema de identificação automática de veículos será objeto de estudo deste trabalho, SIAV [SOU2000], que objetiva melhorar a sua taxa de reconhecimento de placas de identificação, reconhecia somente 37,4% das placas, assim como melhorar também seu tempo de execução.

4.2 Aplicações

As aplicações de um SIAV são muito variadas. Neste item serão abordadas as mais comuns.

Medição e monitoramento do tráfego

A medição e o monitoramento de várias características de viagens e fluxo de tráfego urbano, por meio da tecnologia de vídeo, vem sendo cada vez mais usados para o planejamento de transporte e gerenciamento de tráfego [SHU51]. Estima-se que para o acompanhamento adequado do tráfego em uma auto-estrada é necessário que o sistema utilizado forneça sua resposta em menos de um segundo [BAR99].

A possibilidade de se determinar padrões de movimento pela comparação de pares de placas de licença, ao longo de uma malha rodoviária, levou à primeira implementação de um sistema de identificação automática de veículos em 1990 na Inglaterra. A praticidade deste tipo de sistema foi confirmada em uma série extensiva de tentativas conduzidas em 1993 pela Volpe National Transportation Systems Center (VMTSC) e seus associados [SHU51].

As maiores vantagens deste tipo de sistema, para medição e monitoramento de tráfego, são:

- Possibilidade de fornecimento de grandes quantidades de amostras durante o período de coleta de dados;
- Possibilidade de fornecimento de uma estimação representativa dos tempos de viagem através de amostragem aleatória;
- Fornecimento de tempos de viagem em pequenos intervalos de tempo, fornecendo um perfil de velocidade para estudo durante os horários de pico;

Pagamento de pedágio

Um sistema capaz de reconhecer placas de licença pode ser usado para identificar veículos em praças de pedágios. Isto pode ser feito de duas formas. Na primeira, o sistema pode ser usado em conjunto com um banco de dados contendo os dados de registro do veículo e informações de seu proprietário, para debitar automaticamente a tarifa do pedágio. Este procedimento pode reduzir drasticamente os custos da concessionária da rodovia através da redução do número de pessoas necessárias no local, principalmente em momentos de tráfego intenso, diminuindo ainda o tempo de espera em postos de pedágio.

Na segunda forma este tipo de sistema pode ser usado como sistema de segurança objetivando identificar veículos infratores. Por exemplo, na Itália, uma auto estrada controlada por uma companhia privada possui um sistema de identificação de veículos através de sensoriamento remoto chamado de "Telepass". Este sistema permite que carros portadores de um dispositivo especial, transitem em uma via específica sem necessidade de parar nas praças de pedágio. Entretanto, motoristas fraudulentos podem tentar transitar por esta via evitando o pagamento de pedágio. Nestes casos, um sistema de identificação por imagem pode ser usado para coibir este tipo de comportamento.

Acesso a áreas restritas

O sistema pode ser usado para identificar o abuso em qualquer situação em que o tráfego é restrito. Este é um problema de segurança que pode ser resolvido com o uso de um SIAV.

4.3 Tratamento das Imagens

Pré-processamento

O pré-processamento tem como principais objetivos a correção e a preparação da imagem.

A correção é realizada quando a imagem adquirida apresenta algum problema, como por exemplo: inclinação, ruído ou falhas. Para cada um dos casos são usados algoritmos específicos como ajuste da inclinação, eliminação do ruído e recuperação de falhas.

A preparação é realizada com o objetivo de facilitar o processamento da fase seguinte que é a segmentação. Na preparação, são usados algoritmos para realçar dados

de interesse e eliminar dados que podem dificultar a segmentação. Os procedimentos mais utilizados na preparação são a limiarização, exclusão de linhas e a suavização.

Segmentação

Os algoritmos de segmentação procuram extrair os objetos de interesse da imagem que, posteriormente, serão caracterizados representando uma quantificação da sua forma.

A segmentação dos caracteres envolve a separação individual dos caracteres do resto do conteúdo da imagem digitalizada. Numa primeira etapa de segmentação torna-se importante realizar a diferenciação e separação entre imagens gráficas e imagens textuais. Após, as áreas que foram identificadas como sendo áreas cujo conteúdo são textos sofrem um processamento visando a isolar cada caractere em uma pequena matriz de pontos.

4.4 Sistema Siav 1.0

O projeto de um sistema de visão artificial envolve diversas etapas e, consequentemente, diversas decisões. Estas decisões vão desde o número de bits necessários para representação da imagem até o tipo de técnica de reconhecimento que será empregada no sistema. Para que sejam coerentes durante o desenvolvimento de um sistema deste porte diversas soluções devem ser exaustivamente testadas, conforme foi realizado, o que demanda uma grande quantidade de tempo e trabalho.

O SIAV 1.0[SOU2000] é composto por oito. As cinco primeiras etapas são responsáveis pela localização da placa dentro da imagem, e as três etapas restantes pela extração e reconhecimento dos caracteres nela contidos. A Figura 4.8 apresenta, em ordem, todos os estágios que compõem a primeira versão do sistema.

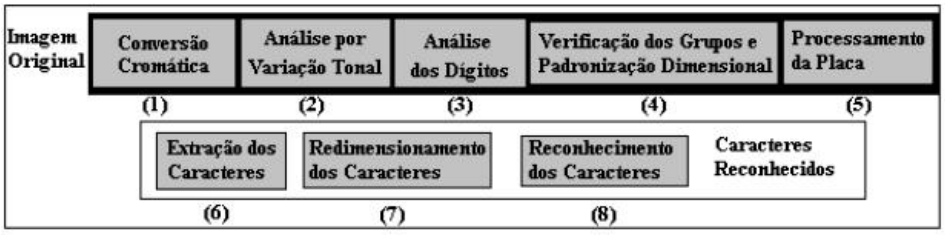


FIGURA 4.8 - Etapas do SIAV 1.0

4.4.1 Localização das Placas

As cinco primeiras etapas do sistema são responsáveis pela localização da placa dentro da imagem, e as três etapas restantes pela extração e reconhecimento dos caracteres nela contidos.

Inicialmente, partindo da imagem original, tem-se a primeira etapa que é a conversão da imagem colorida para monocromática.

A imagem adquirida pelo dispositivo de captura de vídeo é normalmente colorida e representada por 24 bits por pixel. Os algoritmos utilizados para localização e segmentação dos caracteres não fazem uso da informação de cor presente na imagem, pelo contrário, buscam padrões similares e descontinuidades entre pixels em uma imagem monocromática. Por este motivo, se faz necessária uma conversão da imagem original colorida para o tipo monocromática (256 tons de cinza).

A conversão é feita através da análise dos valores das três componentes, vermelho, verde e azul, formadoras de cada pixel da imagem colorida. O pixel resultante *P'(x,y)* na imagem monocromática pode ser calculado através da comparação dos valores das componentes de um pixel na imagem colorida, em busca dos valores mínimo e máximo, conforme a equação 7 [SOU2000].

$$P'(x,y) = \frac{V_{mín} + V_{máx}}{2} \qquad (7)$$

Esta conversão é o resultado parcial de uma conversão do sistema de cores RGB (red, green, blue) para o sistema HSL (hue, saturation, luminance). Ela é dita parcial porque nos fornece apenas uma das componentes do sistema HSL, a componente L de luminância. Esta componente é responsável pela informação de intensidade presente em uma imagem.

A partir da imagem monocromática inicia-se a etapa de localização da placa. Esta etapa é composta por quatro estágios. O primeiro é o algoritmo de análise por variação tonal que faz uso de uma característica construtiva da placa [BAR99], o contraste entre o fundo e os caracteres, para localizá-la. Esta técnica baseia-se no fato de que as linhas onde a placa está localizada na imagem possuem uma clara "assinatura" que faz usualmente possível distingui-las das linhas restantes da imagem. Este procedimento é intuitivo visto que se baseia na existência de caracteres contrastando com um fundo, uma característica obrigatória em qualquer placa de licença. Os elementos contrastantes podem ser localizados através de uma análise por descontinuidades. No caso da placa ser detectada neste estágio, sua localização é armazenada para processamento no próximo estágio.

De acordo com SOUZA[SOU2000], uma análise no desempenho deste algoritmo mostra sua eficiência onde a distribuição de luz é homogênea.

Em casos onde a distribuição de luz sobre a superfície da placa não é homogênea, e ainda, em imagens onde há elementos complexos como paralelepípedos, reflexos, etc., utilizou-se o algoritmo proposto por Coetzee,Botha,Weber[COE98] que mostrou-se mais eficiente, mas um problema em relação a sua implementação é o tempo de processamento. Este estágio consiste de três etapas. Na primeira temos uma binarização local adaptativa sobre a imagem inteira ou sobre o resultado conseguido no estágio anterior. Na segunda etapa um algoritmo de localização de dígitos é utilizados sobre a imagem resultante da etapa de binarização, localizando elementos que possuam dimensões compatíveis com um padrão esperado. E finalmente, na terceira etapa, temos a utilização de um algoritmo de análise dos dígitos encontrados. Nesta etapa é verificada a existência de algum grupo de dígitos com as propriedades espaciais de uma placa de licença; distâncias relativas entre caracteres e dimensões dos mesmos. A seguir é realizada a verificação do número de caracteres e padronização dimensional, ou seja, o sistema verifica se o grupo com probabilidade de ser uma placa possui seis ou sete caracteres e padroniza suas dimensões se necessário. É realizado ainda o processamento da região da placa visando ressaltar as diferenças entre caracteres e o fundo da placa utilizando-se para isto a técnica *contrast stretch*. As características originais são preservadas, permitindo uma segmentação adequada dos caracteres.

4.4.2 Extração dos Caracteres

Uma vez que o local da placa tenha sido encontrado, realçado e definidas suas dimensões finais, é necessário extrair os caracteres do restante da imagem. Isto é feito através da utilização de um algoritmo de agrupamento por similaridade, que tem a finalidade de separar cada elemento encontrado, e de um algoritmo de avaliação dos elementos extraídos, para descartar possíveis ruídos [SOU2000].

4.4.3 Reconhecimento dos Caracteres

Os elementos encontrados e analisados com sucesso na etapa anterior devem ser redimensionados para apresentação ao sistema de reconhecimento baseado em redes neurais, pois as redes utilizadas, *feedforward*, apresentam um número fixo de neurônios na camada de entrada. Por este motivo, é necessário que o vetor de entrada possua uma dimensão fixa, igual ao número de neurônios da camada de entrada para o correto funcionamento da rede.

Uma vez que os caracteres encontram-se redimensionados, os três primeiros são apresentados à rede responsável pelo reconhecimento das letras, e os quatro caracteres

restantes são apresentados à rede responsável pelo reconhecimento dos números. Ambas as redes, nesta primeira versão, possuem 255 neurônios na camada de entrada, o que corresponde a um caracter de dimensões 15 x 15 pixels mais as duas linhas de descrição, como pode ser observado na Figura 4.9.

```
****0000000****
***000000000***
**00000000000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**0000***0000**
**00000000000**
***000000000***
****0000000****
111222222222111
001111222111100
```

FIGURA 4.9 - Caracter 0 (15 x 15)

A rede dedicada ao reconhecimento dos números possui 10 neurônios na camada de escondida e 10 neurônios na camada de saída. A rede para reconhecimento de letras possui 26 neurônios nas camadas escondida e de saída.

4.5 Aperfeiçoamento do Sistema

A solução aqui apresentada visa reconhecer os caracteres da placa de um veículo segundo a atual legislação brasileira de trânsito, com sete caracteres (3 letras e 4 números). Alguns melhoramentos foram realizados no sistema inicial obtendo-se uma melhor taxa de reconhecimento das placas dos veículos, assim como um menor tempo de processamento do sistema. Para verificar a eficiência dos aperfeiçoamentos realizados utilizou-se um conjunto de 750 imagens adquiridas em diferentes oportunidades, com diferentes equipamentos e características. Essas imagens estão divididas em três bancos distintos. O primeiro deles é composto por 250 imagens de automóveis capturadas em diferentes condições de iluminação por uma câmera CCD, este banco foi utilizado para os testes do SIAV 1.0 [SOU2000]. O segundo banco, também composto por 250 imagens de automóveis em movimento, e o terceiro banco são imagens fornecidas pela PROCEMPA, e é constituído de imagens de automóveis infratores, capturadas por pardais, pertencentes ao banco desta empresa.

4.5.1 Redução da Área de Busca

Realizou-se um estudo estatístico com o objetivo de reduzir o tempo de processamento necessário à procura da placa com a menor perda de informação.

Assim, antes da primeira etapa do SIAV1.0 foi incluída uma nova etapa, onde, partindo da imagem inicial capturada, busca-se reduzir á área a ser pesquisada para a procura da placa considerando que existem locais na imagem onde a quantidade de placa é mínima, por exemplo, nos cantos superiores da imagem.

As coordenadas (x, y) dos limites do retângulo da placa foram consideradas variáveis aleatórias. Deste modo, pode-se restringir a área da imagem onde a procura da placa será realizada. Esta procura ocorrerá, então, sobre os pixels de maior probabilidade.

- Captura-se o ponto inicial (esquerdo superior) e final (direito inferior) da placa.
- Calcula-se a média dos pontos obtidos.
- Calcula-se o desvio padrão, tomando-se um desvio padrão a partir da média tem-se que 68,2 % dos dados (placas) estão nesta região.

A Figura 4.10 apresenta o gráfico da distribuição das placas em relação ao número de padrões partindo da média.

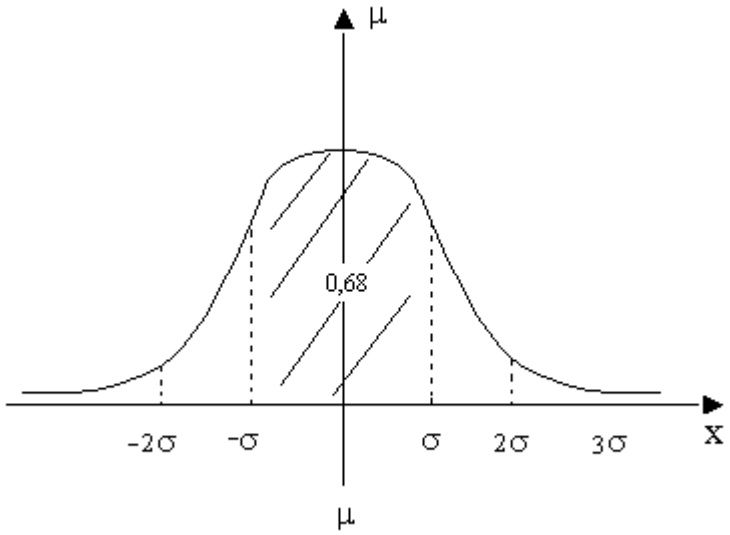


FIGURA 4.10 - Gráfico da Distribuição das Placas em Relação a Média

Numericamente obtém-se:

$Pr\{| x - \mu | \leq \sigma\} \approx 0.682$; $Pr\{| x - \mu | \leq 2\sigma\} \approx 0.954$; $Pr\{| x - \mu | \leq 3\sigma\} \approx 0.997$

Onde x é o ponto, μ é a média e σ o desvio padrão. Assim, se tomarmos dois desvios padrões além da média temos 95,4% das placas nesta região, três desvios padrões a partir da média abrange 99,7% das placas.

Foram analisados os tamanhos de placas para o conjunto de 450 amostras, sendo 150 de cada banco disponível. A análise da distribuição das placas apresentou o seguinte resultado numérico em relação à redução da área de busca da placa.

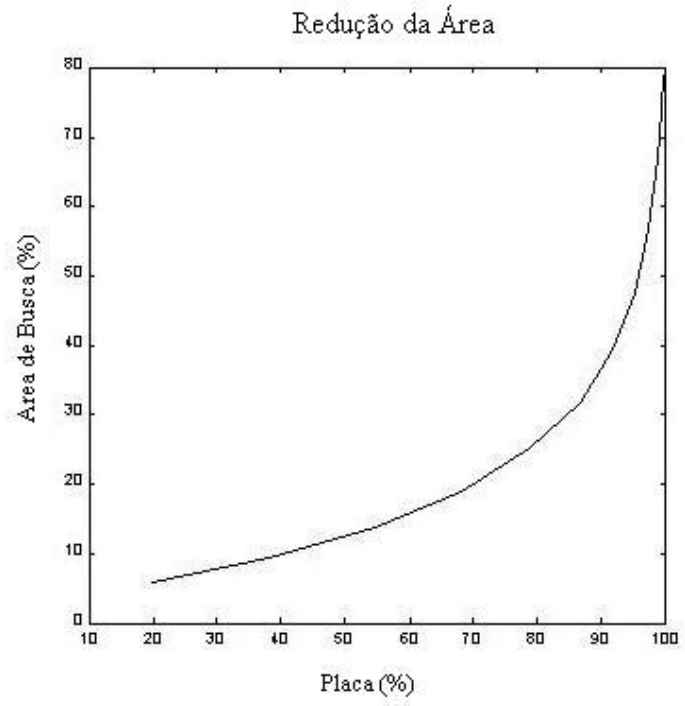


FIGURA 4.11 - Gráfico da Redução da Área de Busca

Neste trabalho, de acordo com a distribuição utilizada, tem-se que 99,7%¨das placas estão concentradas em uma região que possui apenas 60% da área total da imagem. A figura 4.12 apresenta o resultado da redução de uma imagem pertencente ao terceiro banco de teste .

FIGURA 4.12 - Exemplo da área reduzida

Esta imagem representa o pior caso da distribuição, onde não há uma redução no eixo x, relacionado á velocidade do carro, a redução ocorre somente no eixo y, relacionado á posição da câmera.

4.5.2 Pré-processamento da Imagem

A técnica de realce de contraste tem por objetivo melhorar a qualidade das imagens sob os critérios subjetivos do olho humano. É normalmente utilizada como uma etapa de pré-processamento para sistemas de reconhecimento de padrões.

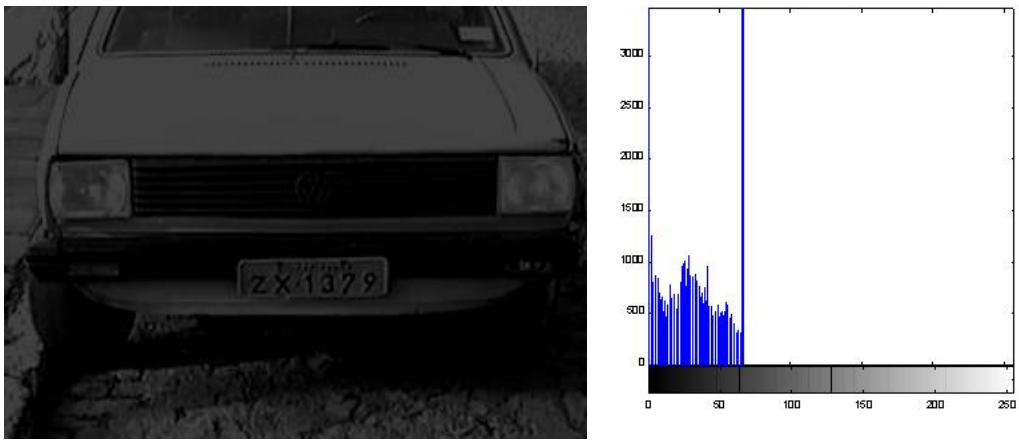
O contraste entre dois objetos pode ser definido como a razão entre os seus níveis de cinza médios.

O objetivo principal das técnicas de realce é processar uma imagem, de modo que o resultado seja mais apropriado para uma aplicação específica do que a imagem original. Para melhorar o contraste das imagens tratadas pelo sistema algumas técnicas de pré-processamento foram implementadas, entre elas a equalização do histograma e algumas técnicas de filtragem .

4.5.2.1 Equalização do Histograma

Técnicas de realce da imagem foram utilizadas neste trabalho objetivando a melhora das imagens para o tratamento em questão. Utilizou-se a equalização do histograma, conforme foi descrito na seção 2.3.3.

O histograma mostrado na figura 4.13(b) representa a probabilidade de ocorrência dos níveis de cinza contida na imagem da figura 4.13(a), presente no banco 2, e mostra que os níveis de cinza estão concentrados em direção a extremidade escura. Assim, este histograma corresponde a uma imagem com características predominantemente escuras.



(a) (b)

FIGURA 4.13 - Imagem com baixo contraste (a) e seu histograma (b)

Para melhor adequar esta imagem à visualização, uma alteração no contraste se faz necessária. Logo, uma modificação no histograma é feita para melhorar a imagem. Na figura 4.14 é apresentado o resultado obtido com a equalização do histograma.

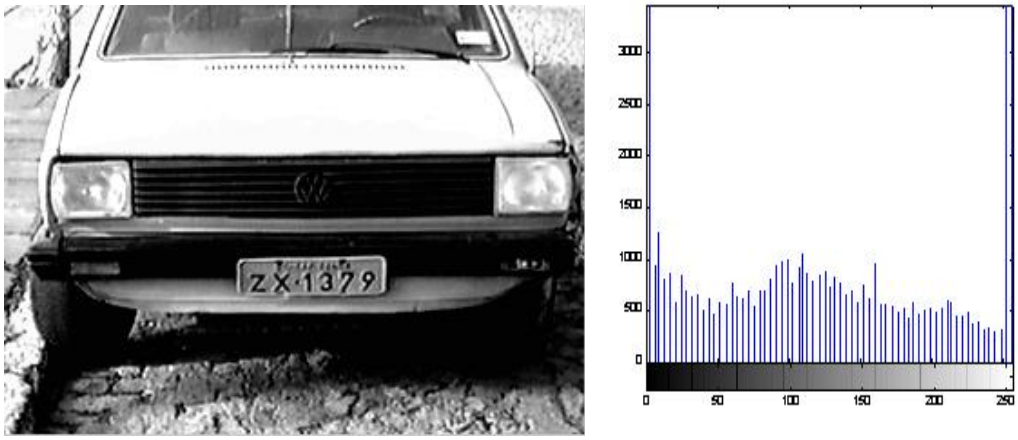


FIGURA 4.14 - Imagem Equalizada (a) e seu histograma (b)

É importante salientar que na imagem original toda informação está contida dentro da faixa mostrada no histograma. A equalização apenas permitiu que esta informação se tornasse mais explícita, sem alterá-la, preservando a proporção de contraste entre os pixels da imagem.

4.5.2.2 Filtragem Homormófica

As imagens pertencentes aos bancos de testes foram capturadas sem uma iluminação controlada, ou seja, com distribuição de luz homogênea e heterogênea sobre a superfície da placa. Imagens com sombras ou reflexos foram tratadas, com bons resultados, por outra técnica de realce implementada. Na filtragem homomórfica [GON93], o modelo de iluminação - reflectância é utilizado como base para este método que busca melhorar a aparência da imagem através da compressão do intervalo de brilho e realce de contraste simultâneos, conforme descrito na seção 2.3.3.3.1. A aproximação de problemas de filtragem não linear através do princípio da superposição generalizada tem apresentado sucesso em algumas aplicações práticas.

Nesta técnica a formação da imagem pode ser modelada como um processo multiplicativo, consistindo basicamente do produto de sinais, um referente ao nível de iluminação incidente e o outro referente à reflectância dos elementos da imagem. A abordagem de realce utilizado é apresentado na figura 4.15 e consiste em dada a imagem de entrada, aplica-se o logaritmo sobre esta imagem e obtém-se a transformada de Fourier de $ln(f(x, y))$. Aplica-se então um filtro $H(u,v)$ [GON93] sobre a transformada do logaritmo da imagem para ter a imagem filtrada.

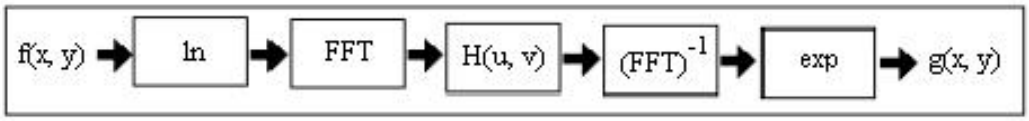


FIGURA 4.15 - Abordagem da Filtragem Homomórfica

O efeito que o processo tem sobre a aparência dos objetos em uma cena é independente da intensidade de luz incidente sobre os mesmos. Similarmente, o efeito que o processo tem sobre a intensidade de luz incidente sobre os objetos é independente da natureza dos mesmos.

Para que possa ser projetado um filtro linear para tratar de forma independente as componentes referentes ao logaritmo da luminância e ao logaritmo da reflectância, tais componentes devem possuir comportamento distinto em freqüência. Na prática, a componente referente à luminância varia lentamente no espaço, enquanto que a componente referente à reflectância é às vezes estática e às vezes dinâmica, devido à textura, tamanho e contorno dos objetos. Assim, somente um processamento parcialmente independente é possível.

A figura 4.16 apresenta uma imagem com reflexo e sombras. No sistema inicial a placa na imagem abaixo não é localizada. Com a inclusão da etapa de pré-processamento a placa é localizada e os caracteres reconhecidos corretamente.

FIGURA 4.16 - Imagem Original e Imagem filtrada com filtro Homomórfico

O filtro utilizado foi o filtro passa baixa de *Butterworth* dado por:

$$H_{(u,v)} = \frac{1}{1 + 0{,}414\left[D(u,v)/Do)\right]^{2n}}$$

Em que D(u,v) é definido por: $D(u,v) = (u^2 + v^2)^{1/2}$
e D_0 é a distância de corte medida a partir da origem do plano da freqüência.

O filtro é de 6° ordem e freqüência de corte igual a 7.

4.5.2.3 Filtragem Butterworth

As imagens pertencentes ao terceiro banco, adquirido por *pardais*, possuem pouco contraste, assim necessitam de uma etapa de pré-processamento. Para realçar as imagens pertencentes a este banco, utilizou-se um filtro passa - baixa de *Butterworth*. A função de transferência deste filtro, de ordem *n* e com uma freqüência de corte posicionada a uma distância igual a 7 da origem é definida pela equação 8.

$$H_{(u,v)} = \frac{1}{1 + 0{,}414\left[D(u,v)/Do\right)\right]^{2n}} \qquad (8)$$

A seguir algumas imagens do terceiro banco são apresentadas, assim como a versão filtrada destas imagens.

FIGURA 4.17 - Imagem Original e Imagem filtrada com Butterworth (8)

FIGURA 4.18 - Imagem Original e Imagem filtrada com Butterworth (8)

Os resultados obtidos com a etapa de pré-processamento serão apresentados no capitulo 6.

4.5.3 Análise da Tonalidade dos Dígitos

Uma das alterações realizadas no sistema original foi a inclusão de uma função de análise de tons dos dígitos presentes em uma placa, ou seja, antes de ser efetivado um grupo válido a tonalidade dos prováveis dígitos é analisada. Os dígitos que possuem tons de cinza semelhantes são agrupados. Assim, o algoritmo de análise dos dígitos encontrados leva em conta além das características dimensionais e posições relativas, o tom do dígito. Este procedimento evita, na maioria dos casos, que o sistema assuma possíveis bordas das placas como caracteres, o que ocorria causando algum engano em relação à extração dos caracteres das placas.

O algoritmo original para localização dos dígitos tem como resultado os elementos encontrados na placa, que serão posteriormente analisados para formar, ou não, um grupo válido.

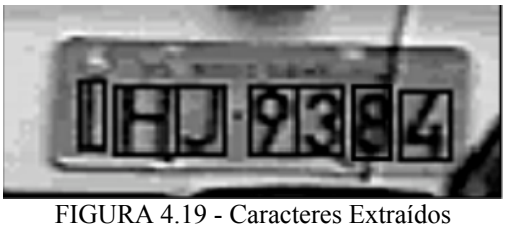


FIGURA 4.19 - Caracteres Extraídos

A alteração realizada no algoritmo original, para formação de um grupo válido, consiste em analisar cada um dos dígitos encontrados verificando a média dos tons de cinza deste com o possível grupo. Se a diferença entre as médias for menor que um determinado limiar então é possível que este seja um grupo válido, dependendo então dos outros quatro critérios de avaliação.

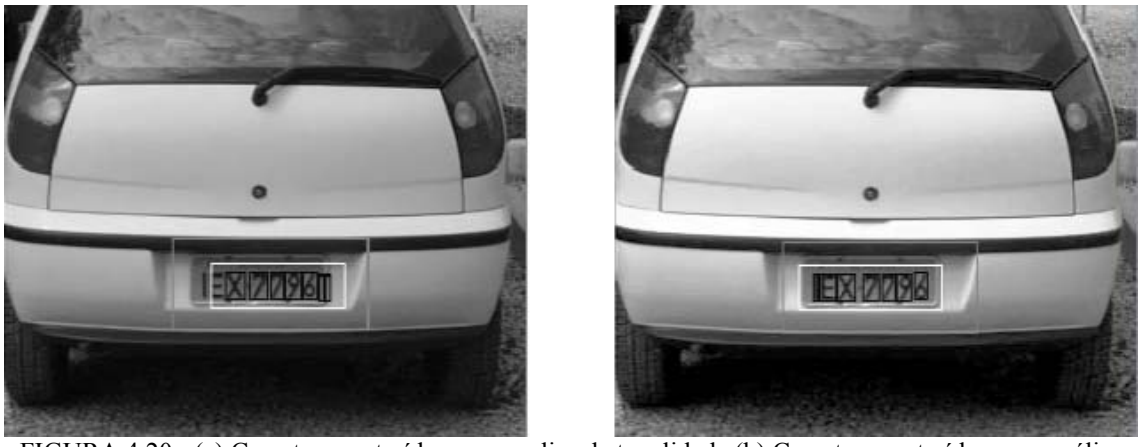


FIGURA 4.20 - (a) Caracteres extraídos sem analise da tonalidade (b) Caracteres extraídos com análise das tonalidades.

4.5.4 Rede Neural

O módulo de reconhecimento é implementado através de uma rede neural. A definição da rede neural mais apropriada a determinado problema é de difícil definição

[HAY94]. Na escolha do número de camadas, definiu-se a camada de entrada, a intermediária e a de saída como estritamente necessárias. As redes utilizadas na primeira e segunda versão do SIAV diferem em alguns pontos. A primeira camada recebe diretamente cada pixel da máscara do caracter extraído, que é uma matriz de 15 x 15 mais duas linhas de descrição, a camada intermediária difere nas duas versões, tanto em número de neurônios quanto em função de ativação utilizada. A última camada totaliza e emite os resultados.

Tendo sido definidos o número de camadas e a quantidade de neurônios em cada camada, 255 x 10 x 26 para rede de letras e 255 x 10 x 10 para rede de números, foi escolhida a função de ativação utilizada no sistema. Diferente do SIAV 1, a função de ativação escolhida para os neurônios da camada intermediária foi a tangente hiperbólica e para os neurônios da camada de saída a função de ativação usada é a logística. Estas alterações na rede, juntamente com um treinamento mais apropriado, resultaram em uma melhor taxa e reconhecimento do sistema.

4.5.4.1 Seleção de Amostras

Inicialmente deve-se selecionar amostras representativas das classes de interesse e então utilizá-las para o treinamento da rede. A rede neural dedicada ao reconhecimento de números foi treinada com 20 amostras de cada caracter, sendo estas selecionadas entre um banco de imagens específico para este fim, ou seja, amostras selecionadas a partir dos caracteres não reconhecidos pertencentes as imagens dos três bancos utilizados. O processo de treinamento consistiu na apresentação das 20 amostras de cada caracter, um caracter por vez, até a rede convergir para um erro quadrático médio menor que 2,6e-5.

A rede neural dedicada ao reconhecimento das letras foi treinada com 20 amostras para cada letra, obedecendo o mesmo processo acima.

Várias outras redes neurais foram treinadas porém, os resultados foram inferiores aos apresentados pelas redes descritas acima.

4.5.4.2 Treinamento da Rede

Para realizar o treinamento da rede neural *feedforward* foi escolhido o método *backpropagation*. Este método foi implementado no software Matlab. A rede neural foi treinada a partir do arquivo de treinamento baseado nas amostras selecionadas na etapa anterior. Treinar a rede de acordo com este algoritmo de treinamento significa fornecer os valores de entrada e modificar os pesos conforme a saída desejada.

O treinamento da rede ocorre em etapas denominadas épocas. Estas épocas consistem em se aplicar todos os dados de entrada de treinamento na rede, verificar o erro de cada um desses dados e ajustar a rede para diminuir o erro médio. Isto evita que, ao se ajustar a rede para uma entrada, se aumente os erros das outras.

4.5.4.3 Simulação da Rede

Após a obtenção da rede neural treinada, surge a necessidade de verificar se os resultados obtidos foram satisfatórios, ou seja, se a rede neural conseguiu definir uma regra suficientemente geral para reconhecer corretamente a quantidade de placas não só das imagens utilizadas para o treinamento mas também das imagens que lhe são até então desconhecidas.

Os resultados obtidos são apresentados no capítulo 6.

4.5.4.4 Topologia da Rede

Uma vez que os caracteres encontram-se redimensionados, os três primeiros são apresentados à rede responsável pelo reconhecimento das letras enquanto os quatro últimos são apresentados à rede responsável pelo reconhecimento dos números. Ambas as redes possuem 255 neurônios na camada de entrada.

A rede dedicada ao reconhecimento dos números possui 10 neurônios da camada escondida e 10 neurônios na camada de saída. A rede para o reconhecimento das letras possui também 10 neurônios nas camadas escondidas e 26 neurônios na camada de saída, como é mostrado na figura 4.21.

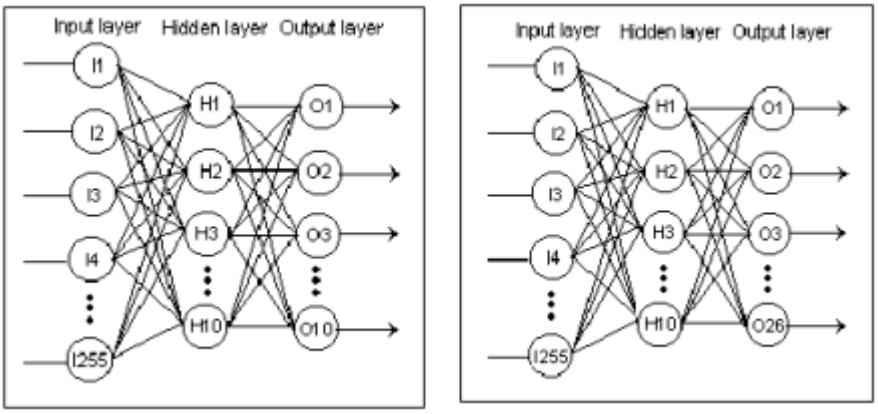


FIGURA 4.21 - Redes Neurais para Números e Letras

A saída da rede rotula o padrão de entrada com o neurônio de resposta mais elevada [SOU2000]. Em outras palavras, uma vez que um determinado vetor de entrada é apresentado à rede, o neurônio da camada de saída que possuir a maior resposta

indicará a que classe aquele vetor pertence. No caso da rede dedicada aos números, a associação do número reconhecido é diretamente ligada ao neurônio vencedor (0 a 9). Na rede para reconhecimento de letras, cada neurônio da camada de saída está associado a uma letra do alfabeto como pode ser visto na tabela abaixo.

TABELA 4.2 - Associação dos Neurônios da Camada de Saída

A	*0*	*N*	*13*
B	1	O	14
C	2	P	15
D	3	Q	16
E	4	R	17
F	5	S	18
G	6	T	19
H	7	U	20
I	8	V	21
J	9	W	22
K	10	X	23
L	11	Y	24
M	12	Z	25

5 Resultados

5.1 Introdução

Neste capítulo são apresentados os resultados obtidos com a utilização das técnicas descritas na seção 4.5 sobre o SIAV 1.0 desenvolvido em [SOU2000]. Para verificar a eficiência das alterações realizadas foram utilizadas três bancos de imagens, todos compostos por 250 imagens, num total de 750 imagens, adquiridas em diferentes oportunidades, com diferentes equipamentos e características. Devido ao grande número de imagens, apenas algumas são mostradas em detalhes.

Uma comparação do desempenho do novo sistema em relação ao sistema anterior [SOU2000], assim como a comparação com um software demonstrativo [HIG99] desenvolvido por uma empresa israelense, disponível na Internet, também é apresentada.

Um computador PC AMD K6-2 500MHz com 64 MBRAM foi utilizado para a obtenção dos resultados apresentados neste capítulo.

5.2 Siav 2.0

Nesta versão do SIAV as seguintes operações estão implementadas:

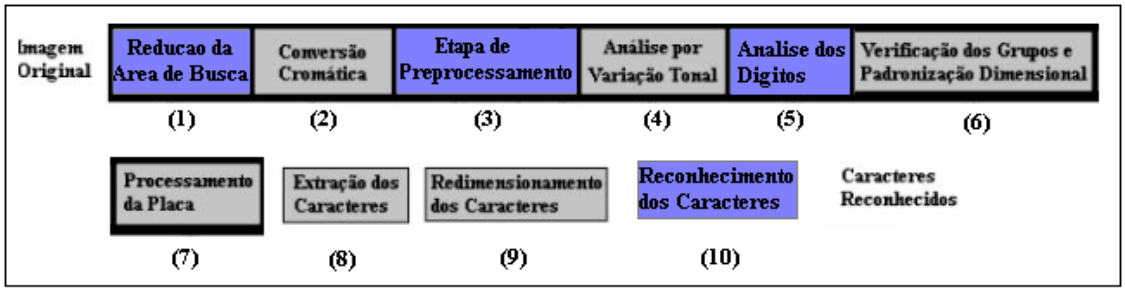


FIGURA 5.1 – Etapas do SIAV 2.0

Na versão SIAV 2.0 dois módulos foram incluídos, um responsável pela redução da área de busca e outro que realiza o pré-processamento da imagem. As técnicas de pré-processamento implementadas são a equalização do histograma, a filtragem homomórfica e a filtragem Butterworth. Além disso, outros dois estágios foram alterados, no estágio responsável pela análise dos dígitos foi incluída uma rotina para análise do tom dos dígitos pertencentes a um mesmo grupo. No estágio de reconhecimento dos caracteres a rede neural foi alterada e retreinada conforme descrito anteriormente.

As operações citadas acima não são implementadas no SIAV 1.0 e são utilizadas automaticamente na versão SIAV 2.0, resultando em um melhor desempenho.

5.3 Estudo Estatístico

Conforme apresentado na seção 4.5.1, pode-se restringir a área da imagem onde a procura da placa será realizada. Esta procura ocorrerá, então, sobre uma janela de pixels da imagem na qual há maior probabilidade de localizar a placa.

Foram analisados os tamanhos de placas para o conjunto de 150 amostras para cada banco. A análise da distribuição das áreas de placas apresentou o resultado numérico em relação à redução da área de busca da placa apresentado na tabela 5.1.

TABELA 5.1 - Redução da Área de Busca

Área de Busca (%)	*Placas (%)*		
	Banco1	Banco2	Banco3
10	93,3	38	33,3
20	98	54,6	46,6
30	98,6	76,6	68
40	99,3	92	80,6
50	100	96	86
60	100	97,3	88,6
70	100	98,6	93,3
80	100	99,3	98

Nesta tabela, a área de busca representa a área a ser pesquisa em busca da placa, e placas representa a porcentagem de placas presentes nesta área sobre o total (100%) de placas, ou seja, em 10% da área total concentram-se 93,3% das placas do banco1, 38% das placas do banco2 e 33,3% das placas do banco3. De acordo com a linha 6 da tabela tem-se 100% das placas do banco1, 96% das placas do banco2 e 86% das placas do banco3 estão concentradas em uma região que possuí apenas 60% da área total da imagem.

Pode-se notar, de acordo com os dados apresentados, que para o primeiro banco de imagens a redução da área de pesquisa pode ser feita de forma mais acentuada com menor perda. Isto se deve ao fato de a captura da imagem ter sido feita com uma câmera operada por um fotógrafo que realiza o enquadramento manualmente do veículo na janela da imagem capturada. Inicialmente acreditava-se que com o banco de imagens de veículos capturadas em condições controladas, como as obtidas de veículos em situação de infração, os resultados seriam ainda melhores devido a maior padronização do enquadramento do veículo nas mesmas. Isto não se confirmou, como pode ser

observado nos dados apresentados na Tabela 5.1, pois o banco 3 de imagens foi capturado por "pardal" automático instalado em via urbana de Porto Alegre.

5.4 Pré-Processamento da Imagem

A partir do sistema inicial foram implementadas três técnicas de realce de imagem visando melhorar a taxa de reconhecimento de caracteres. O objetivo principal das técnicas de realce é processar uma imagem de modo a melhorá-la para uma aplicação específica [GON93].

5.4.1 Equalização do Histograma

A etapa de pré-processamento modifica e prepara os valores dos pixels de uma imagem com o objetivo de facilitar para que as operações subsequentes alcancem melhores resultados [AWC96]. Uma das mais simples e mais importantes técnicas de realce é a equalização do histograma. Como foi mencionado anteriormente, o histograma de uma imagem representa as freqüências relativas de ocorrência de vários níveis de cinza de uma imagem [JAI89]. Histogramas mal distribuídos, concentrados em um ponto, apresentam imagens ruins (muito claras ou muito escuras). A equalização do histograma de uma imagem serve para melhor distribuir (uniformizar) os valores dos níveis de cinza de tal forma que se obtenha uma imagem com mais qualidade.

Os resultados obtidos com a equalização do histograma nas imagens pertencentes aos três bancos são apresentadas a seguir.

FIGURA 5.2 - Imagem Original e Imagem Equalizada pertencente ao Banco1

FIGURA 5.3 - Imagem Original e Imagem Equalizada pertencente ao Banco2

FIGURA 5.4 - Imagem Original e Imagem Equalizada pertencente ao Banco3

O algoritmo de equalização do histograma foi testado em 750 imagens diferentes, 250 pertencentes a cada banco, e apresentou uma melhora de 20,8% no reconhecimento total do sistema com um tempo médio de processamento de 0.6 segundos para o primeiro banco, que possui uma resolução de 320 x 240 pixels por imagem, a 5 segundos para o terceiro banco onde as imagens possuem resolução 640 x 480. A figura 5.5 apresenta a resposta do sistema com a etapa de equalização implementada.

FIGURA 5.5 - Processamento da Imagem Original e da Imagem Equalizada respectivamente

Na figura acima pode-se observar a resposta do sistema original, onde a placa não foi reconhecida corretamente, pois houve perda de informação no caracter G. Com a imagem pré-processada, utilizando-se a técnica de equalização do histograma, a placa foi reconhecida corretamente.

5.4.2 Filtragem Homomórfica

Outra técnica de realce de imagem implementada foi a filtragem homomórfica, onde o modelo de iluminação – reflectância. Reflectância *r(x, y)* é a quantidade de luz refletida pelos objetos da cena e iluminação *i(x, y)* é a quantidade de luz incididente na cena, é utilizado como base para este método que busca melhorar a aparência da imagem através de compressão do intervalo de brilho e realce de contraste simultâneos.

Inicialmente é feita uma conversão logarítmica sobre a imagem, seguida do cálculo da transformada da mesma e da filtragem passa-banda e reconversão FFT e exponenciação.

Esta técnica de filtragem é indicada para cenas ao ar livre onde a iluminação é variável. A filtragem homomórfica foi realizada em 750 imagens diferentes, 250 pertencentes a cada banco, e apresentou uma melhora de 34,9% no reconhecimento total do sistema com um tempo médio de processamento de 0.75 segundos por imagem, no pior caso, para o primeiro banco a 6,2 segundos por imagem, para o pior caso, do terceiro banco. A figura 5.6 apresenta a resposta do sistema com a etapa de filtragem implementada.

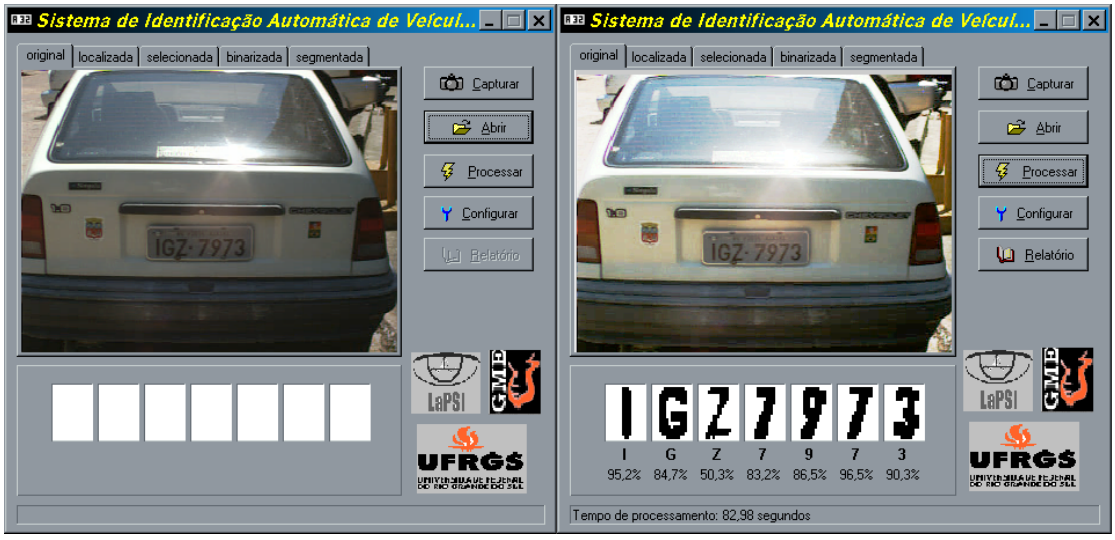


FIGURA 5.6 – Processamento da Imagem Original e da Imagem Filtrada respectivamente

Na figura 5.6, observa-se que inicialmente a placa não era localizada na imagem, após a filtragem a placa é localizada e reconhecida corretamente.

5.4.3 Filtro Butterworth.

As imagens pertencentes ao terceiro banco, adquiridas por *pardais*, possuem pouco contraste, Assim, necessitam de uma etapa de pré-processamento que realce o contraste presente na imagem. Para realçar as imagens pertencentes a este banco, utilizou-se um filtro passa - alta de *Butterworth*. A função de transferência deste filtro, de ordem *n* e com uma freqüência de corte posicionada a uma distância 7 da origem é definida pela equação 9.

$$H_{(u,v)} = \frac{1}{1+\left[fr / fc)\right]^{2n}} \qquad (9)$$

A seguir é apresentado o resultado obtido com a utilização da filtragem.

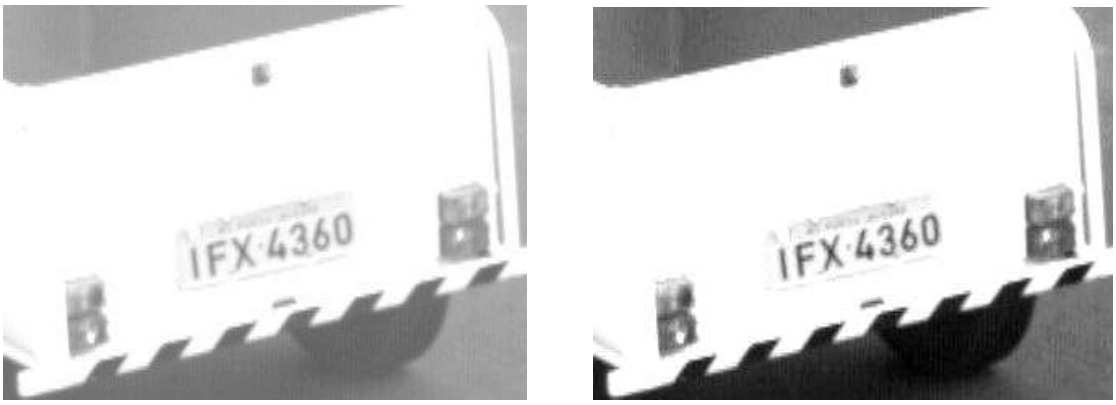


FIGURA 5.7 – Imagem Original e Imagem Filtrada pelo filtro Butterworth de ordem n

5.5 Análise da Tonalidade dos Dígitos

Uma das alterações realizadas no sistema original foi a inclusão de uma função de análise de tons dos dígitos presentes em uma placa, ou seja, antes de ser efetivado um grupo válido a tonalidade dos prováveis dígitos é analisada.

O algoritmo original para localização dos dígitos tem como resultado os elementos encontrados na placa, que serão posteriormente analisados para formar, ou não, um grupo válido como apresentado na figura 5.8.

FIGURA 5.8 – Resultado da Etapa de Extração

A alteração realizada no algoritmo original, para a formação de um grupo válido, consiste em analisar cada um dos dígitos encontrados verificando a média dos tons de cinza deste com o possível grupo. Se a diferença entre as médias for menor que um determinado limiar então é possível que este seja um grupo válido, dependendo então dos outros quatro critérios de avaliação. O limiar é determinado calculando-se a média μ dos pixels pertencentes à região segmentada.

FIGURA 5.9 – Resultado da Etapa de Extração do SIAV1 e do SIAV2

A figura 5.9 apresenta os resultados obtidos com o sistema original e com o sistema alterado. Pode-se observar que no sistema original havia erro na etapa de extração dos caracteres, o que foi corrigido com a alteração realizada.

5.6 Rede Neural

Redes neurais são modelos de processamento capazes de organizar em classes um determinado conjunto de padrões de entrada. Ao se apresentar um valor à entrada de uma rede neural, esta fornece na saída um resultado que indica a classe a que pertence este valor. Para que possa classificar de forma correta as entradas, a rede neural precisa ser treinada em um processo iterativo. Tal processo consiste na apresentação à rede de sucessivas entradas para que esta se adapte à função a ser desempenhada.

Devido á baixa taxa de acerto no reconhecimento da placa de licença de veículos apresentado pelo sistema SIAV 1.0 (37,5%) a rede foi retreinada tendo em vista que, inicialmente, a rede neural dedicada ao reconhecimento de letras foi treinada com apenas cinco amostras para cada letra e a rede neural dedicada ao reconhecimento dos números foi treinada com quinze amostras de cada caracter, sendo estas amostras selecionadas entre um banco de imagens específico para este fim.

Duas redes neurais *feedfoward* utilizando o algoritmo *backpropagation* para treinamento estão sendo utilizado para o reconhecimento. Os três primeiros caracteres são apresentados à rede responsável pelo reconhecimento das letras, e os quatro caracteres restantes são apresentados à rede responsável pelo reconhecimento de números. A figura 5.10 apresenta a topologia das redes.

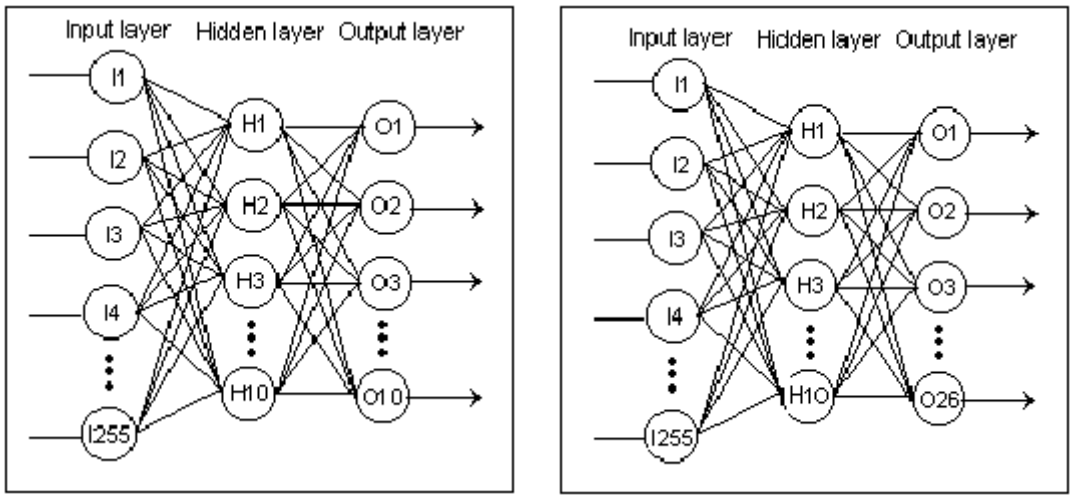


FIGURA 5.10 – Rede para reconhecimento dos números e letras respectivamente

Para o treinamento da rede de letras foram extraídas 15 amostras a partir das imagens que apresentavam problemas no reconhecimento e 5 para a rede de números totalizando 20 amostras de cada caracter para o treinamento. O processo de treinamento constitui na apresentação das 20 amostras de cada caracter, um caracter por vez, até a rede convergir para um erro médio quadrático de 10^{-5}. A figura 5.11 apresenta um exemplo de mapa de bits obtido com a extração de um caracter que é apresentado à rede.

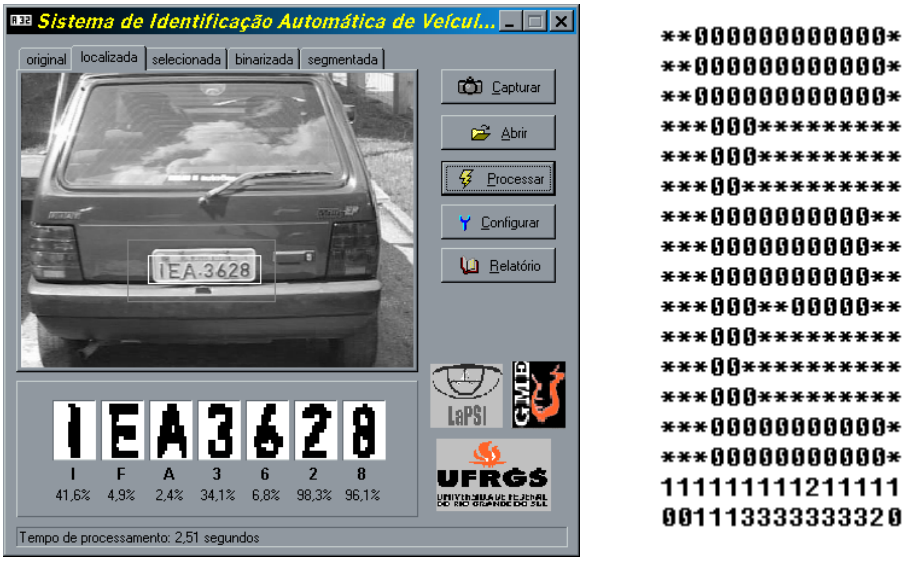


FIGURA 5.11 – Exemplos de extração de caracteres

Para o treinamento da rede neural foi utilizado o software Matlab [MAT96]. Algumas alterações foram realizadas na rede durante a fase de treinamento. A rede de números possui 255 x 10 x 10 neurônios nas camadas de entrada, escondida e de saída, respectivamente, e a rede de letras possui 255 x 10 x 26 neurônios por camada. As funções de ativação também sofreram alterações no sistema SIAV 2.0; na camada intermediária a função de ativação utilizada foi a tangente hiperbólica e na camada de saída a função utilizada foi a logística. No sistema SIAV 1.0 apenas a função de ativação tangente hiperbólica foi utilizada.

A função logística, dada por $g(h) = \frac{1}{1+\exp(-h)}$, é representada pelo gráfico da figura 5.12.

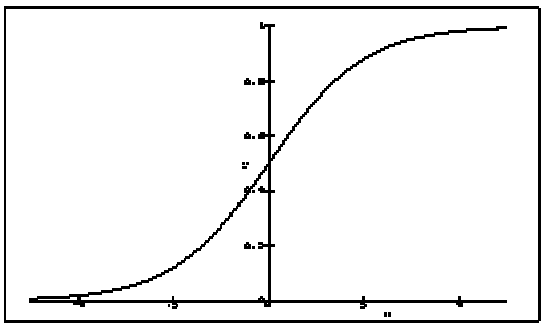


FIGURA 5.12 – Função de Ativação Logística

A função tangente hiperbólica, $g(h) = \tanh(\beta h)$, é apresentada na figura 5.13.

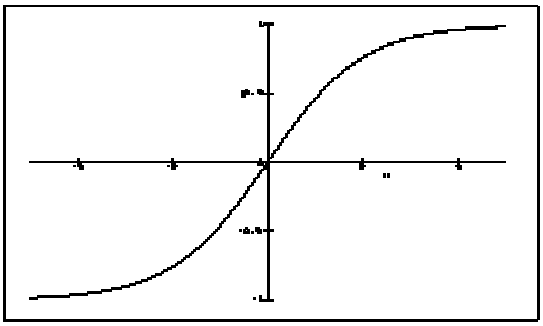


FIGURA 5.13 – Função de Ativação Tangente Hiperbólica

Os resultados das redes treinadas e com as alterações descritas acima, para um conjunto de 450 imagens obtidas a partir dos três bancos de imagens utilizados neste trabalho, são apresentados no gráfico da figura 5.14. O primeiro par de barras refere-se à rede utilizada pela primeira versão do SIAV [SOU2000]. O segundo par de barras refere-se a rede retreinada com as alterações citadas nesta seção. A primeira e maior barra, em cada par de barras, refere-se ao número de caracteres corretamente reconhecidos considerando a totalidade dos caracteres analisados, ou seja, dos 3150 caracteres analisados, 2560 foram reconhecidos utilizando a primeira versão do sistema e 2809 foram reconhecidos com a segunda versão. A segunda barra, em cada par de barras, refere-se ao número de placas corretamente reconhecidas (os sete caracteres presentes na placa corretamente reconhecidos) , ou seja, das 450 placas analisadas 112 foram reconhecidas inicialmente, ou seja, 24,9%. A rede neural utilizada no SIAV 2.0 obteve 45,33 % de sucesso na identificação correta dos sete caracteres contidos na placa (204 placas), o que representou melhoria considerável de desempenho da rede neural de reconhecimento de caracteres, após a realização das alterações explicadas anteriormente. É importante salientar que os dados apresentados no gráfico da figura 5.13 são referentes ao reconhecimento dos caracteres individualmente para a análise da rede neural, e não o reconhecimento das placas presentes em uma imagem, estes dados serão fornecidos mais adiante.

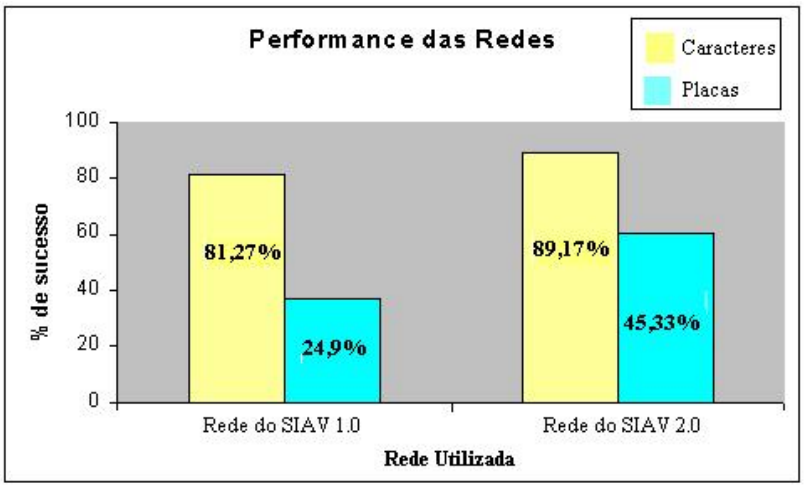


FIGURA 5.14 – Resultados Comparativos das Redes Neurais

5.7 Resultados Gerais e Comparativos

O sistema aperfeiçoado (SIAV 2.0) foi testado nos três bancos, citados anteriormente, num total de 750 imagens e comparado com o sistema original (SIAV 1.0) e com um software demonstrativo - *Seecar* - utilizado como parâmetro de comparação durante o desenvolvimento do sistema inicial (SIAV 1.0). Os resultados obtidos são apresentados nos gráficos abaixo, sendo apresentados os resultados obtidos com os sistemas no banco 1, 2 e 3 respectivamente.

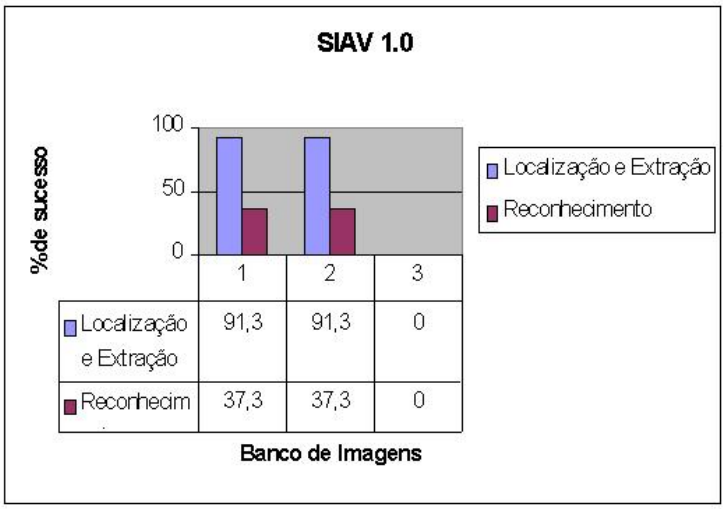


FIGURA 5.15 – Resultados Comparativos no SIAV 1.0

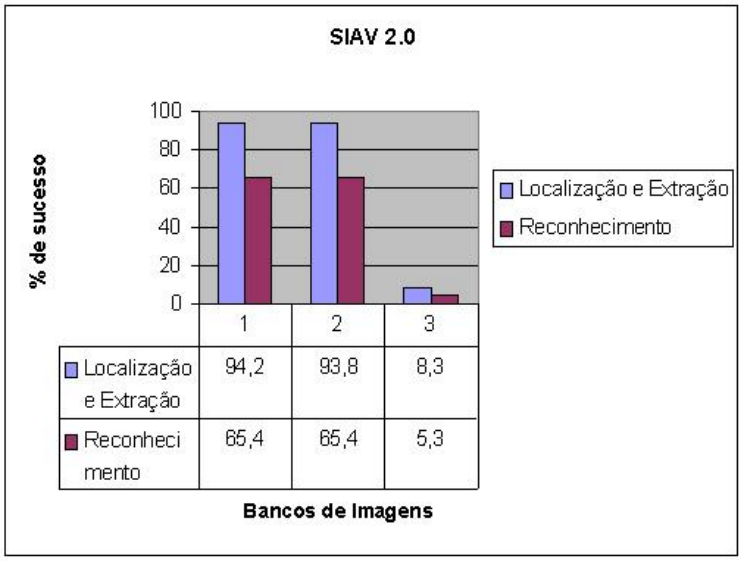


FIGURA 5.16 – Resultados Comparativos no SIAV 2.0

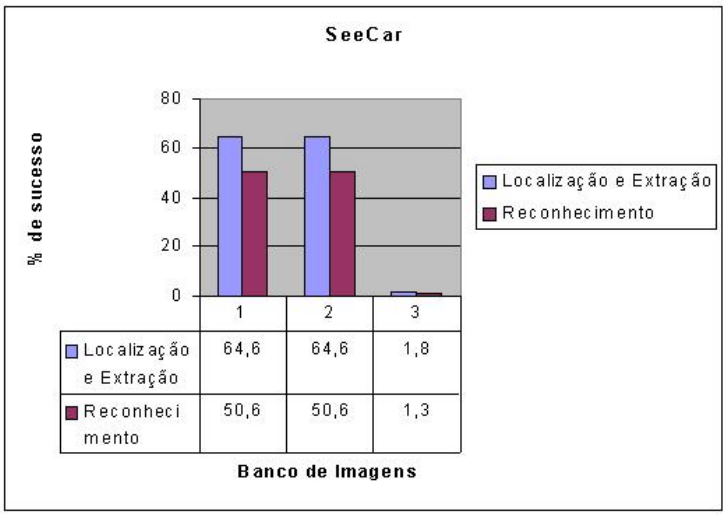


FIGURA 5.17 – Resultados Comparativos no SeeCar

Pode-se salientar as baixas taxas apresentadas pelo banco3. Isto ocorre devido ao fato do trabalho ter sido desenvolvido sobre os dois banco iniciais e o terceiro banco foi obtido somente quando o trabalho já estava boa parte implementado sendo utilizado principalmente para teste. Estas taxas mostram também que é importante para este tipo de aplicação uma análise prévia da imagem a ser trabalhada. Mesmo o sistema não sendo voltado especificamente para imagens do banco três, comparando o SIAV2.0 com os outros dois sistemas ele apresentou melhores taxas de reconhecimento.

6 Implementação de Redes Neurais em Hardware

6.1 Introdução

As formas de implementação de redes neurais são basicamente duas: por software e por hardware.

Na implementação por hardware, o tempo de desenvolvimento de um protótipo é maior comparado ao tempo utilizado em uma implementação em software. Contudo, devido à alta taxa de processamento paralelo que pode ser conseguida pelo hardware, torna-se ideal para aplicações que envolvam o processamento de sinais em tempo real.

As redes neurais artificiais implementadas em hardware possuem uma vantagem em relação ao seu tempo de processamento, em compensação apresentam alguns problemas específicos como, por exemplo, o atendimento à precisão requerida pelo sistema, assim como a implementação de memória específica no sistema neural em hardware. As vantagens e os problemas serão discutidos nesta seção assim como uma arquitetura para o sistema de reconhecimento desenvolvido neste trabalho.

6.2 Considerações Iniciais

Para a implementação em hardware existem várias considerações importantes. Inicialmente é necessário definir as tabelas necessárias ao funcionamento da rede, assim como o tamanho destas.

Uma vez que existe uma relação direta entre o número de neurônios da rede e a capacidade de memória necessária para armazenar pesos, limiares e resultados, o estudo precisa determinar qual a menor rede em número de neurônios que consegue classificar corretamente as letras alvo do sistema.

Uma representação da rede com a indicação dos dados que precisam ser armazenados pode ser vista na figura 6.1.

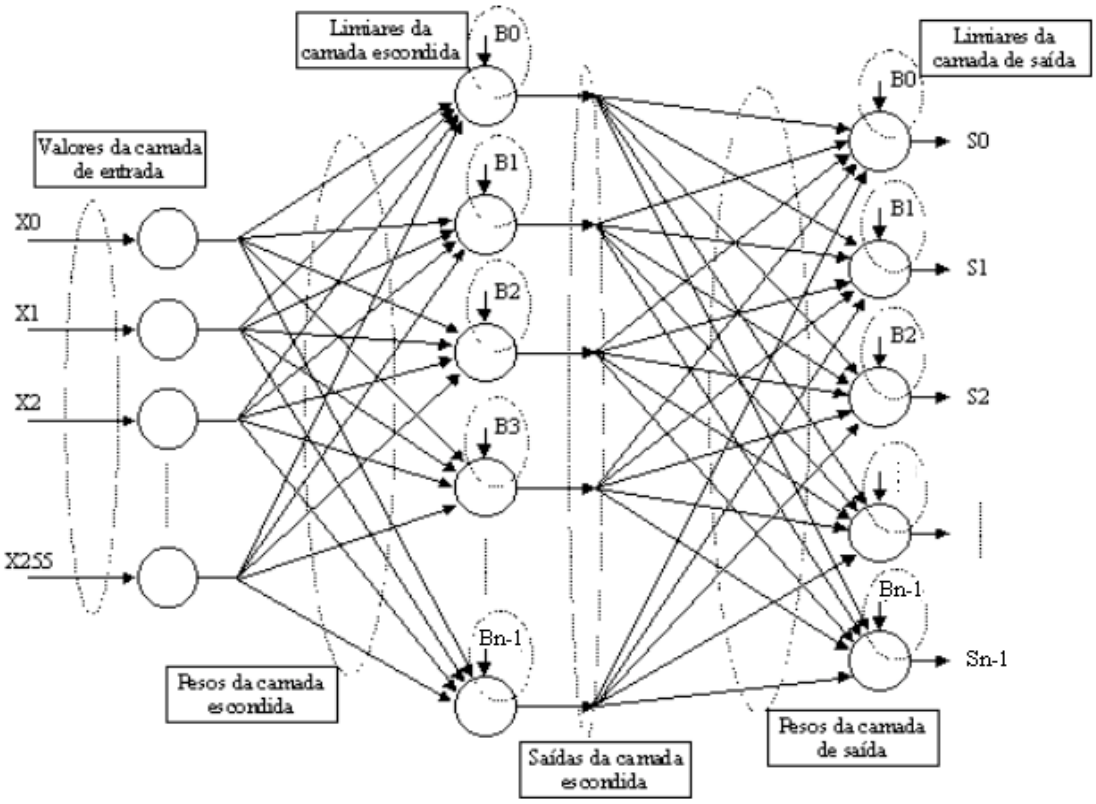


FIGURA 6.1 – Relação de Valores a ser Armazenados em uma Rede Neural

Na prática, as funções de ativação mais utilizadas são as sigmoidais diferenciáveis, representadas pela função logística ([0;1]) e pela tangente hiperbólica ([-1;1]). As funções logística e hiperbólica permitem uma melhor convergência dos valores de saída durante o treinamento, fornecendo melhores resultados durante a operação das redes.

A função de ativação utilizada na camada escondida foi a tangente hiperbólica, representada na figura 6.2. Esta função é implementada em hardware como uma tabela e seu tamanho deve ser definido.

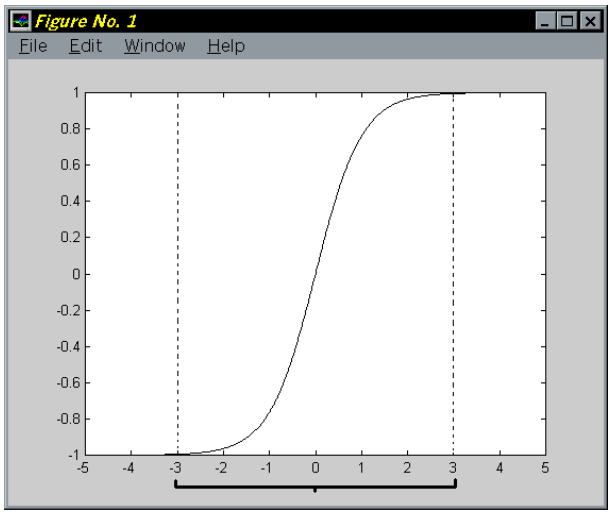


FIGURA 6.2 – Gráfico da Tangente Hiperbólica

De acordo com o gráfico acima pode ser determinado o intervalo no eixo *x* necessário para esta função. O domínio da variável x tem comprimento 6, (3-(-3) = 6). A precisão escolhida implica que o intervalo [-3, 3] deve ser dividido no mínimo em (6*1000) intervalos de tamanhos iguais, isto implica em 13 bits para a representação.

$$4096 = 2^{12} < 6000 < 2^{13} = 8192$$

Assim a tabela necessária para a primeira função de ativação utilizada é demonstrada abaixo:

TABELA 6.1 - Tabela da Função Tangente Hiperbólica

&	θ
$(0000)_H$	-1
$(0001)_H$	-1
.....	.....
$(0FFF)_H$	-0.7616
$(1000)_H$	0
.....	.....
$(1FFE)_H$	1
$(1FFF)_H$	1

A segunda função de ativação utilizada é a função logística que é representada pelo gráfico da figura 6.3.

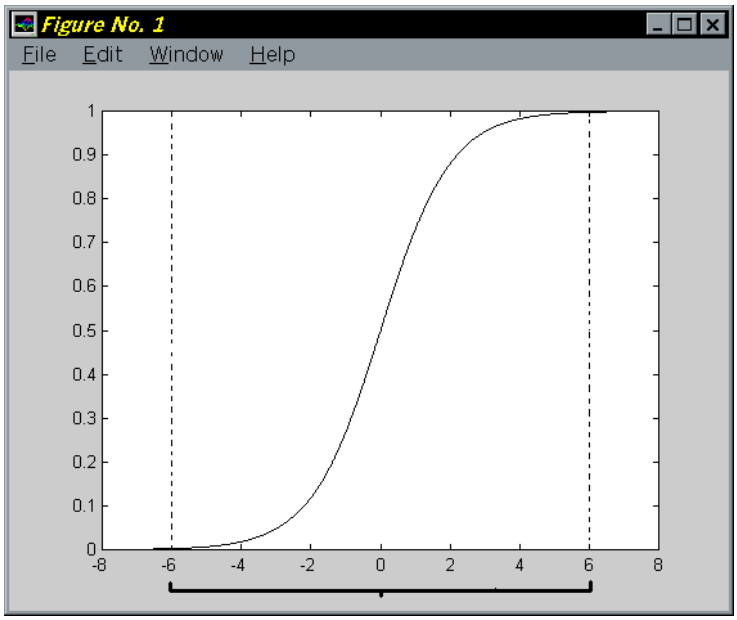


FIGURA 6.3 – Gráfico da Função Logística

De acordo com o gráfico acima pode ser determinado o intervalo para esta função. O domínio da variável x tem comprimento 8, (4-(-4) = 8). A precisão escolhida implica que o intervalo [-4, 4] deve ser dividido no mínimo em (8*1000) espaços de tamanhos iguais, isto implica em 13 bits para a representação.

De acordo com dados obtidos a partir de simulação, esta tabela foi substituída por uma lógica combinacional que será detalhada na seção 6.3 na descrição de uma proposta para a arquitetura da rede neural.

O tamanho das tabelas dos limiares também deve ser definido. Para a determinação do número de bits necessário a representação dos limiares deve-se obter o menor e o maior valor de limiar, para os limiares da camada escondida na rede destinada ao reconhecimento das letras temos, de acordo com os dados obtidos durante a simulação, - 2,7540 e 3,1046 respectivamente como menor e maior valor de limiar, e em seguida aplicar a fórmula abaixo, observando que a variação que ocorre nos limiares após o quarta dígito decimal é insignificante, assim temos:

$B_- = \log2\ [(2,7540*1000)] = 11.42$

$B_+ = \log2\ [(3,1046*1000)] = 11.59$

Tem-se a necessidade de 12bits + sinal para representar os limiares da camada escondida. Esta tabela possui 10 posições tanto para a rede de letras quanto para a rede de números.

A tabela que contém os limiares da camada de saída da rede para reconhecimento dos caracteres contém 10 posições de 13 bits cada. Assim, para o armazenamento dos limiares são necessários 20 posições de 13 bits cada, num total de 260 bits.

Os pesos também são armazenados em tabelas e devem ser analisados para a determinação do tamanho necessário desta. Para a rede de letras, assim como para a de números, o tamanho necessário da tabela que armazena os pesos da camada escondida é de 2560 (256*10) posições com 13 bits cada, como pode ser observado na tabela 6.2.

TABELA 6.2 - Tabela dos Pesos da Camada Escondida

0	*13 bits* .....
1	
2	
.....	
255	
	1 2 10

Pesos da Camada Escondida

Assim, para armazenar os pesos utilizados pela camada escondida são necessários 33280 bits de memória. Para os pesos da camada de saída são necessárias 260 posições de 13 bits cada, para a rede de letras e de 100 posições de 13 bits para a rede de números, como é apresentado na tabela 6.3.

TABELA 6.3 - Tabela dos Pesos da Camada de Saída para a Rede de Letras

0	*13 bits* .....
1	
2	
.....	
9	
	1 2 26

Pesos da Camada de Saída

Uma vez que o número de neurônios da entrada e da saída depende diretamente do problema (256 entradas e 26 neurônios respectivamente para a rede de letras e 256 entradas e 10 para a rede de números), as alterações na rede somente podem ocorrer a nível de neurônios da camada escondida. A tabela 6.4 apresenta a relação entre o tamanho da rede e a capacidade de memória necessária para a implementação da rede responsável pelo reconhecimento dos caracteres alfabéticos e a tabela 6.5 representa o tamanho necessário em bits para a rede de números.

TABELA 6.4 - Capacidade de Memória Necessária para a Rede para Reconhecimento de Letras

Neurônios	*Número de pesos*		*Número*	*Saídas*	*Função de*	*Capacidade*
Escondida (n)	Escondida	Saída	*Limiares*	Escondida(n)	*Ativação*	*em bits*
10	2560	260	36	10	8192	
	33280	3380	468	130	106496	143754

TABELA 6.5 - Capacidade de Memória Necessária para a Rede para o Reconhecimento dos Números

Neurônios	*Número de pesos*		*Número*	*Saídas*	*Função de*	*Capacidade*
Escondida (n)	Escondida	Saída	*limiares*	Escondida(n)	*Ativação*	*em bits*
10	2560	100	20	10	256	
	33280	1300	260	130	106496	141466

6.3 Arquitetura Da Rede Neural

6.3.1 Introdução

As redes neurais artificiais têm sido muito utilizadas em muitos campos interdisciplinares, tanto para desenvolvimento como aplicação. Um dos estudos interessantes de redes neurais é a implementação destas em *hardware* [ALT2000] .

Esta seção apresenta a proposta de uma arquitetura de rede neural para reconhecimento de caracteres. As características desta rede foram apresentadas nas seções anteriores. Serão abordados os blocos necessários a arquitetura bem como a sua parte operativa e parte de controle. Os problemas encontrados como a precisão requerida e a necessidade de memória também são tratados.

No caso de redes neurais, torna-se evidente o uso de valores não inteiros. Neste caso, para se evitar a utilização da notação em ponto flutuante, adotou-se como alternativa utilizar os valores dos pesos, limiares e a função de ativação multiplicados por uma constante (cte = 1000), tratando os valores quando necessário.

A primeira parte da rede utiliza apenas um somador do tipo Ripple_Carry para efetuar o processamento da camada escondida, uma vez que cada valor de peso deve ser multiplicado por um valor binário correspondente a um píxel ativado ou não. Desta forma, o multiplicador de números inteiros pode ser substituído por um multiplicador lógico que efetua a operação lógica AND entre o valor do píxel e o valor do peso correspondente. Isto não apenas diminui a área da parte operativa mas também permite uma diminuição do tempo de operação da camada escondida, a qual é responsável pela maior quantidade de processamento.

Um problema a ser considerado é a função de ativação. Esta função, para determinadas topologias de redes neurais, torna a implementação mais difícil e as vezes menos exata. Quanto a estes problemas, pode-se solucioná-lo através de dois processos. O primeiro é a adoção de uma função de ativação que se adeque mais à técnica de

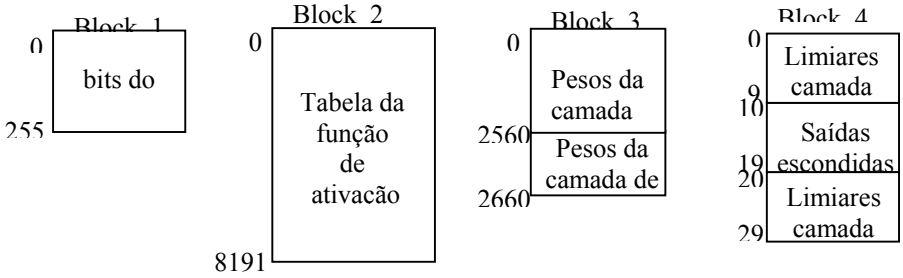
implementação digital. Isto depende da topologia da rede neural bem como o problema que se deseja solucionar. O segundo método, que foi empregado neste trabalho, é a implementação da função de ativação por meios de *lookup tables*. Através deste método, são armazenados previamente valores que serão as respostas para determinadas entradas.

6.3.2 Arquitetura

A quantidade de memória necessária à implementação da rede neural é um fator crítico no sistema. Uma vez que a rede capaz de classificar os caracteres desejados necessita de 143754 bits (tabela 6.4) para as letras, e 141466 bits (tabela 6.5) para os números, para a sua implementação prática, a prototipação da mesma fica comprometida devido às limitações do FPGA disponível (FLEX 10K20 [TRI95]), que possui apenas 12288 bits de memória interna. Para este projeto será necessário o uso do dispositivo APEX 20K. O anexo A traz algumas características deste dispositivo [ALT2000] . Para validação da arquitetura de implementação da rede em VHDL, tem-se algumas soluções. Primeiro, especificar uma rede menor, seguindo os passos e critérios citados anteriormente. Outra solução é implementar as memórias como ROMs externas ao FPGA; neste caso deve-se considerar o tempo de acesso necessário.

Devido aos problemas citados foi implementada uma rede reduzida para o sistema de reconhecimento, os blocos necessários para a implementação forma simulados e estão descritos no anexo B.

Os bancos necessários, para a rede de reconhecimento de números, são apresentados na figura 6.4.



Block_1: 255 x 2 = 512 bits
Block_2: 8192 x 13 = 106496 bits
Block_3: 2660 x 13 = 34580 bits

FIGURA 6.4 – Espaços de endereçamento das memórias da rede

Para gerar os endereços necessários para o seqüenciamento da rede, foi utilizado um circuito contador, responsável por três saídas de endereços. O circuito completo da parte das memórias pode ser visto na figura 6.5.

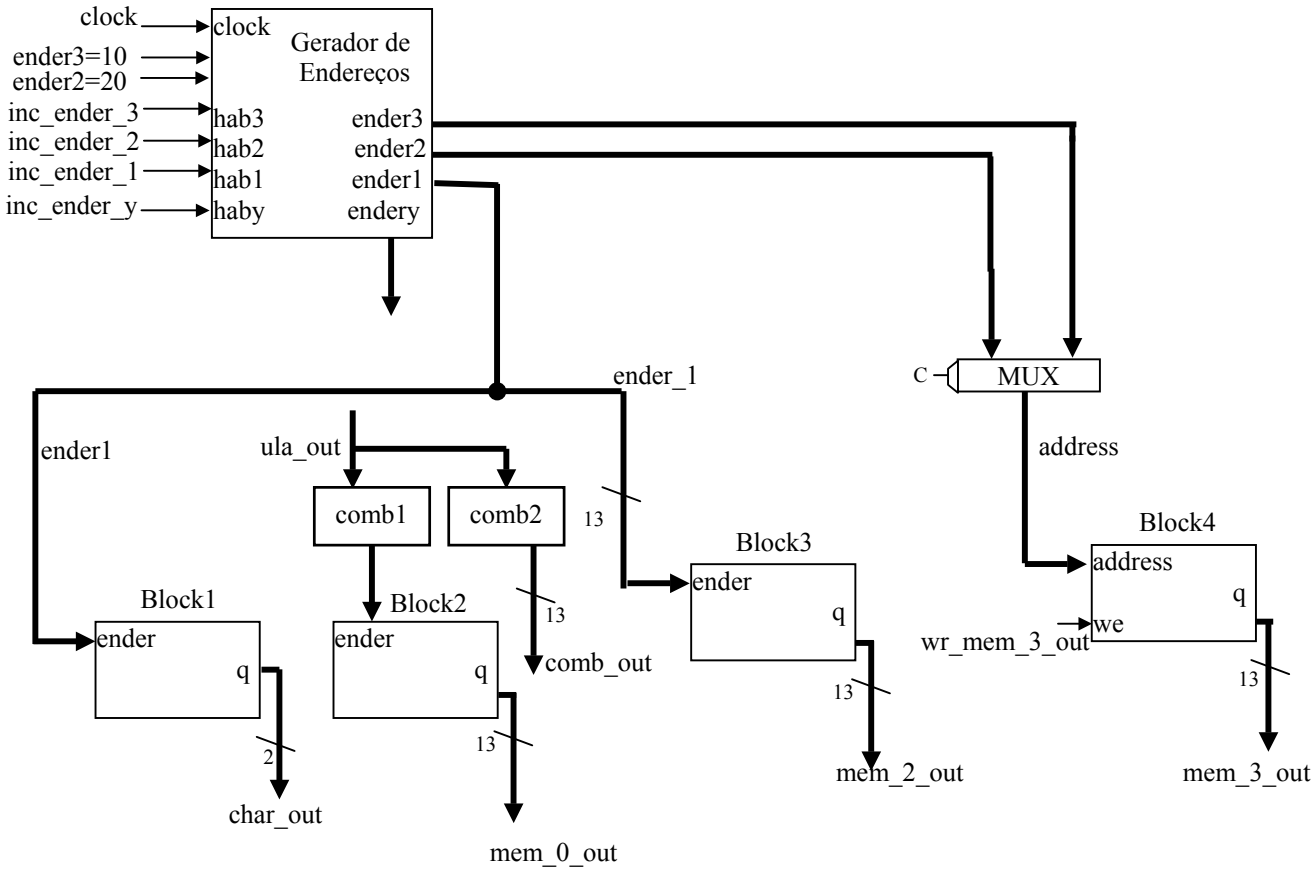


FIGURA 6.5 – Circuito de endereçamento e organização da memória da rede

A função de transferência tangente hiperbólica é implementada através de uma tabela em memória. Isto é conseguido através da utilização do resultado de um neurônio (**ula_out**) como um ponteiro para a memória com os valores da função de ativação.

A figura 6.6 mostra em detalhes a estrutura da parte operativa da rede neural. A parte operativa da rede neural é baseada em um somador do tipo Ripple_Carry de 24 bits e um multiplicador do tipo Booth seqüencial de 12 bits. Uma vez que as multiplicações são realmente necessárias apenas na camada de saída para esta rede, não existe um ganho em termos de desempenho com a utilização de um multiplicador mais rápido, já que a freqüência de operação da rede, com este circuito, sofre uma redução. A não necessidade de multiplicações na camada escondida se deve ao fato das entradas

serem binárias e, portanto, o resultado de uma multiplicação de um valor de peso por uma entrada corresponder a zero ou ao próprio valor do peso. Assim, o multiplicador pode ser substituído por um circuito simples que efetua a multiplicação lógica (operação AND) do peso pela entrada.

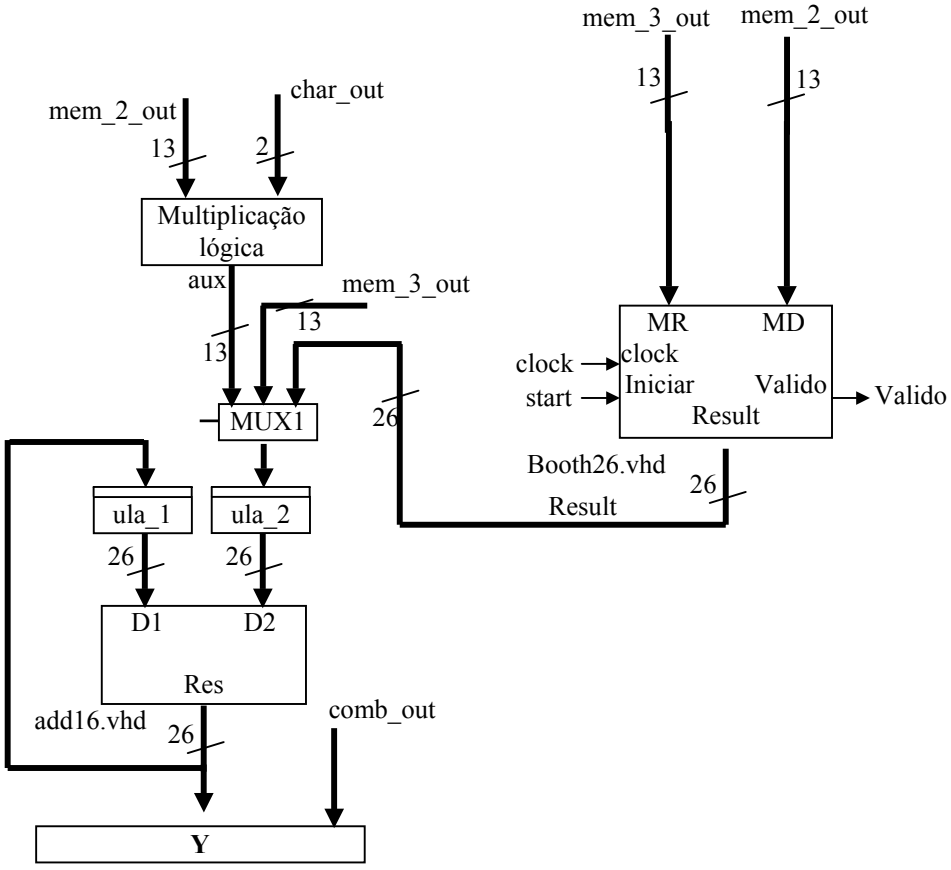


FIGURA 6.6 – Estrutura da parte operativa da rede

O funcionamento da parte de controle da rede neural está baseado em uma máquina de estados finitos com 18 estados, sendo 8 estados responsáveis pelo processamento da camada escondida, 9 estados para o processamento da camada de saída, um estado de espera e um de inicialização. O diagrama de estados da parte de controle da rede é mostrado na figura 6.7.

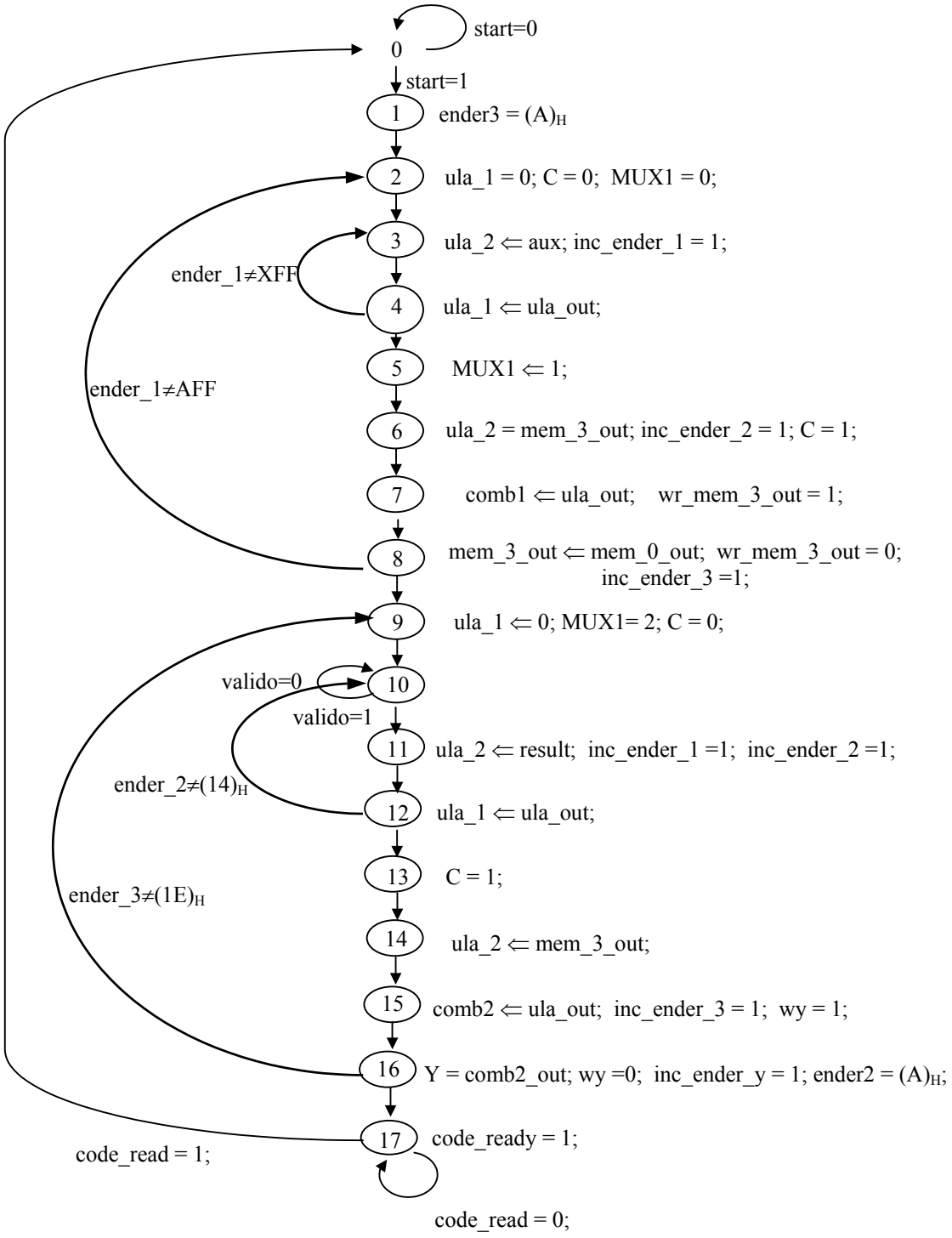


FIGURA 6.7 – Estrutura da parte de controle da rede

6.3.3 Implementação da Rede Neural

Uma vez que a rede desenvolvida em software neste trabalho que classifica os caracteres desejados necessita de 143754 bits para a rede de letras e 141466 para a rede de números para a sua implementação prática, a prototipação da mesma fica comprometida devido às limitações do FPGA disponível (FLEX 10K20), que possui apenas 12288 bits de memória interna. Para validação da arquitetura de implementação da rede em VHDL, , adotou-se como solução, uma rede menor foi especificada e treinada seguindo o passos mostrados anteriormente. O propósito da rede reduzida é a elaboração de um circuito para o reconhecimento de números representados em uma matriz de 32 pontos (8 x 4). A rede reduzida é caracterizada por 32 neurônios na camada de entrada, 8 na camada escondida e 4 na camada de saída.

Inicialmente é importante salientar que a prototipação da rede neural foi desenvolvida somente na forma de uma rede para reconhecimento dos números. Além da facilidade de implementação da rede reduzida, a capacidade de organização da memória interna dos FPGAs foi determinante para a implementação desta versão.

A figura 6.8 corresponde à interface da rede neural. Os sinais de saída corresponde ao código do numeral reconhecido (code_out) e o sinal de código válido (code_ready).

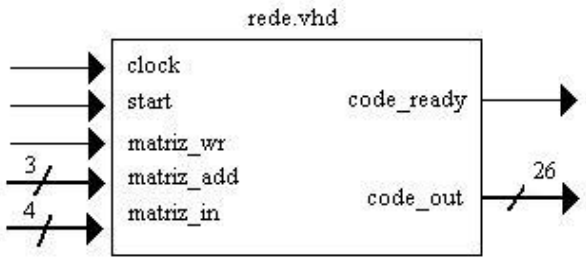
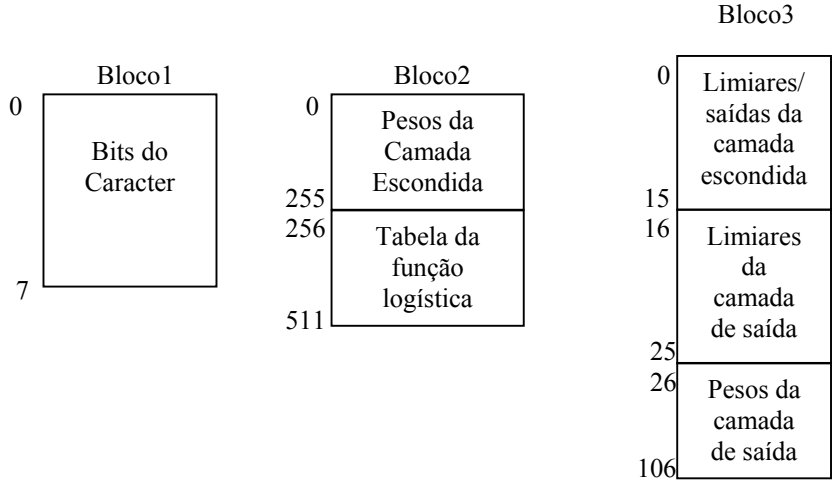


FIGURA 6.8 – I/O da Rede Neural Reduzida para Reconhecimento de Números

A figura 6.8 corresponde à interface de entrada de saída da rede neural. Os sinais de saída correspondem ao código do numeral reconhecido (code_out) e ao sinal de código válido (code_ready).

Para efetuar o processamento correspondente à camada escondida é utilizado apenas um somador do tipo Carry_Select, uma vez que cada valor de peso deve ser multiplicado por um valor binário correspondente a um pixel ativado ou não. O multiplicador de números inteiros pode ser substituído por um multiplicador lógico como foi citado anteriormente, na seção 6.3.1.

A quantidade de memória interna utilizada para a rede reduzida de números é da ordem de 7448 bits. Conforme visto na figura 6.9.



Bloco 1: 8 x 4 = 32 bits

Bloco 2: 512 x 12 = 6144 bits

Bloco 3: 106 x 12 = 1272 bits

FIGURA 6.9 – Espaço de Endereçamento das Memórias da Rede

TABELA 6.6 - Capacidade de Memória Necessária para a Rede Reduzida para o Reconhecimento dos Números

Neurônios	*Número de pesos*		*Número*	*Saídas*	*Função de*	*Capacidade*
Escondida (n)	Escondida	Saída	*limiares*	Escondida(n)	*Ativação*	*em bits*
8	256	80	18	8	256	
	3072	960	216	96	3072	7416

Na figura 6.9 pode-se observar que os valores das saídas da camada escondida e os limiares da camada escondida compartilham um mesmo espaço, sendo que os endereços pares armazenam os limiares e os endereços ímpares armazenam as saídas.

Para gerar os endereços necessários para o seqüenciamento da rede, foi utilizado um circuito contador semelhante ao da figura 6.5. O circuito completo implementado pode ser visto na figura 6.10.

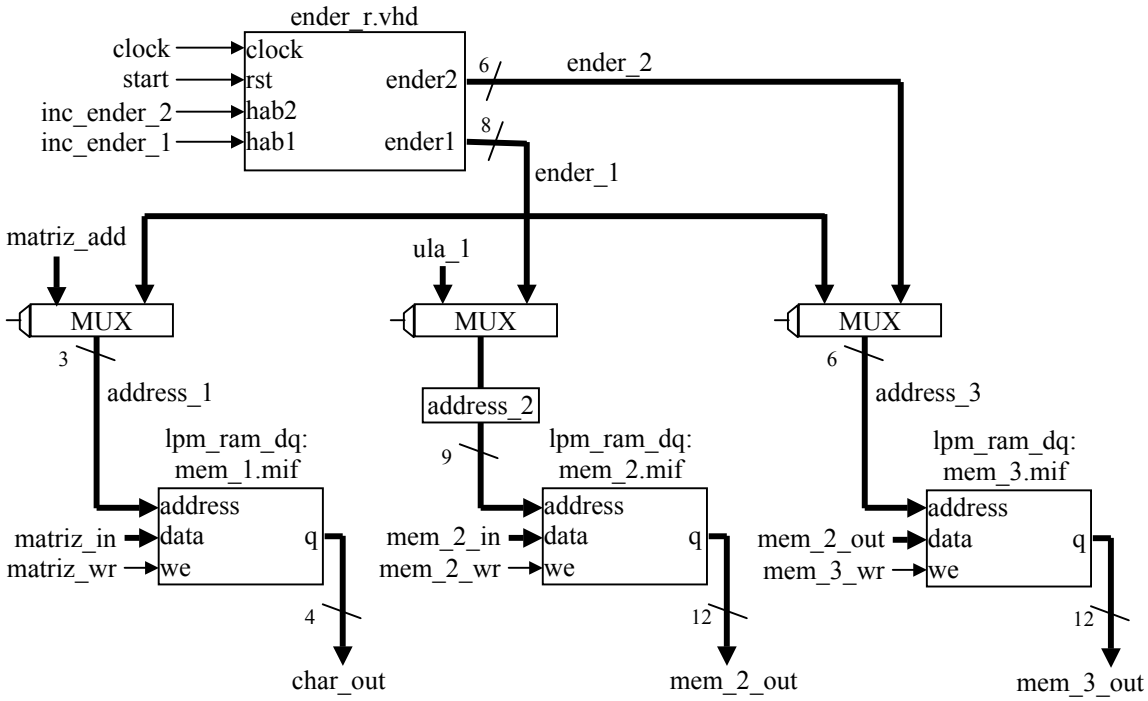


FIGURA 6.10 – Circuito de endereçamento e organização da memória da rede

Os arquivos **.mif** armazenam os dados necessários para a inicialização das memórias da rede e precisam estar presentes na compilação do circuito.

A função de transferência sigmoidal é implementada através de uma tabela em memória [EPP98] . Isto é conseguido através da utilização do resultado de um neurônio (**ula_1**) como um ponteiro para a memória com os valores da função de ativação.

A figura 6.11 mostra em detalhes a estrutura da parte operativa da rede neural, semelhante a discutida na seção 6.3.2. A parte operativa da rede neural é baseada em um somador do tipo Ripple_Carry de 16 bits e um multiplicador do tipo Booth seqüencial de 12 bits [FRA99].

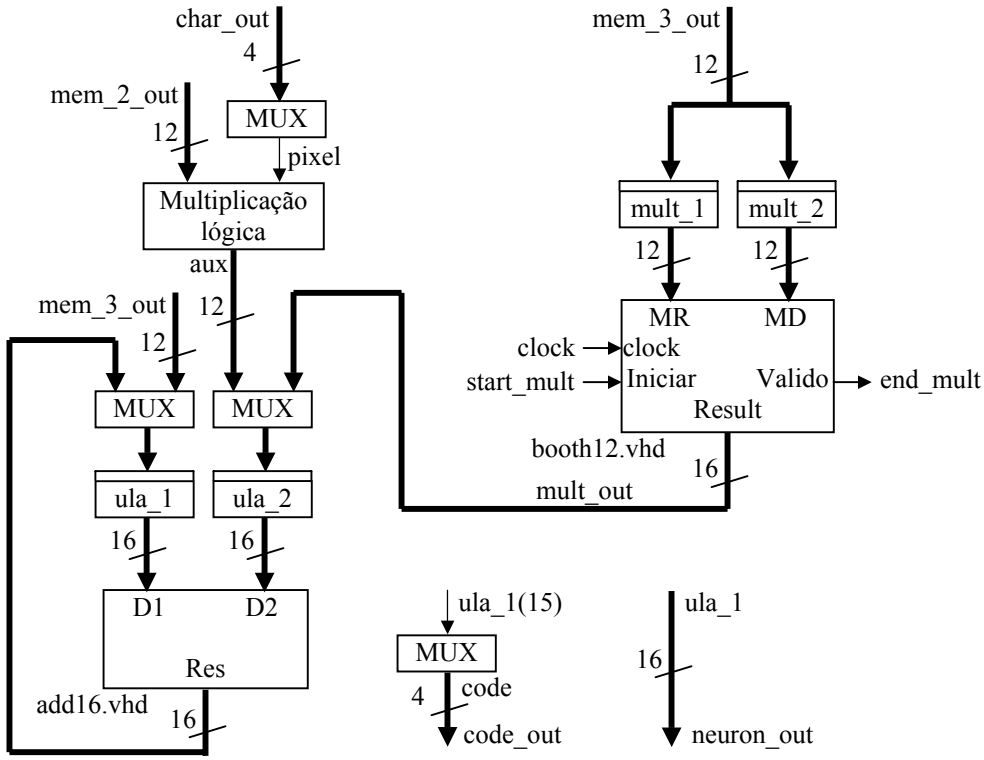


FIGURA 6.11 – Estrutura da Parte Operativa da Rede Reduzida para reconhecimento dos números

O funcionamento da parte de controle da rede neural está baseado em uma máquina de estados finitos com 16 estados, sendo 6 estados responsáveis pelo processamento da camada escondida, 9 estados para o processamento da camada de saída e um estado de espera e inicialização, a máquina de estados também se assemelha a abordada na seção anterior. O diagrama de estados da parte de controle da rede é mostrado na figura 6.12.

É interessante observar que a parte operativa da rede não precisa sofrer modificações muito grandes para sua utilização no reconhecimento de caracteres com um número maior de bits. O principal impacto deste tipo de alteração é na capacidade de memória da rede que precisa aumentar significativamente. Da mesma forma, o gerador de endereços precisa acompanhar o acréscimo no número de posições e a máquina de estados precisa levar em conta os novos endereços.

Com relação ao desempenho de rede, o número de ciclos de relógio necessários para a classificação de um padrão de entrada pode ser determinado pela análise do diagrama de estados, e levando em conta que o multiplicador seqüencial precisa de 7 ciclos de relógio para fornecer um resultado. As equações 6.1 e 6.2 mostram o cálculo do número de ciclos de processamento da rede para um número qualquer de entradas

(ne) e neurônios da camada de saída (cs) e escondida (ce).

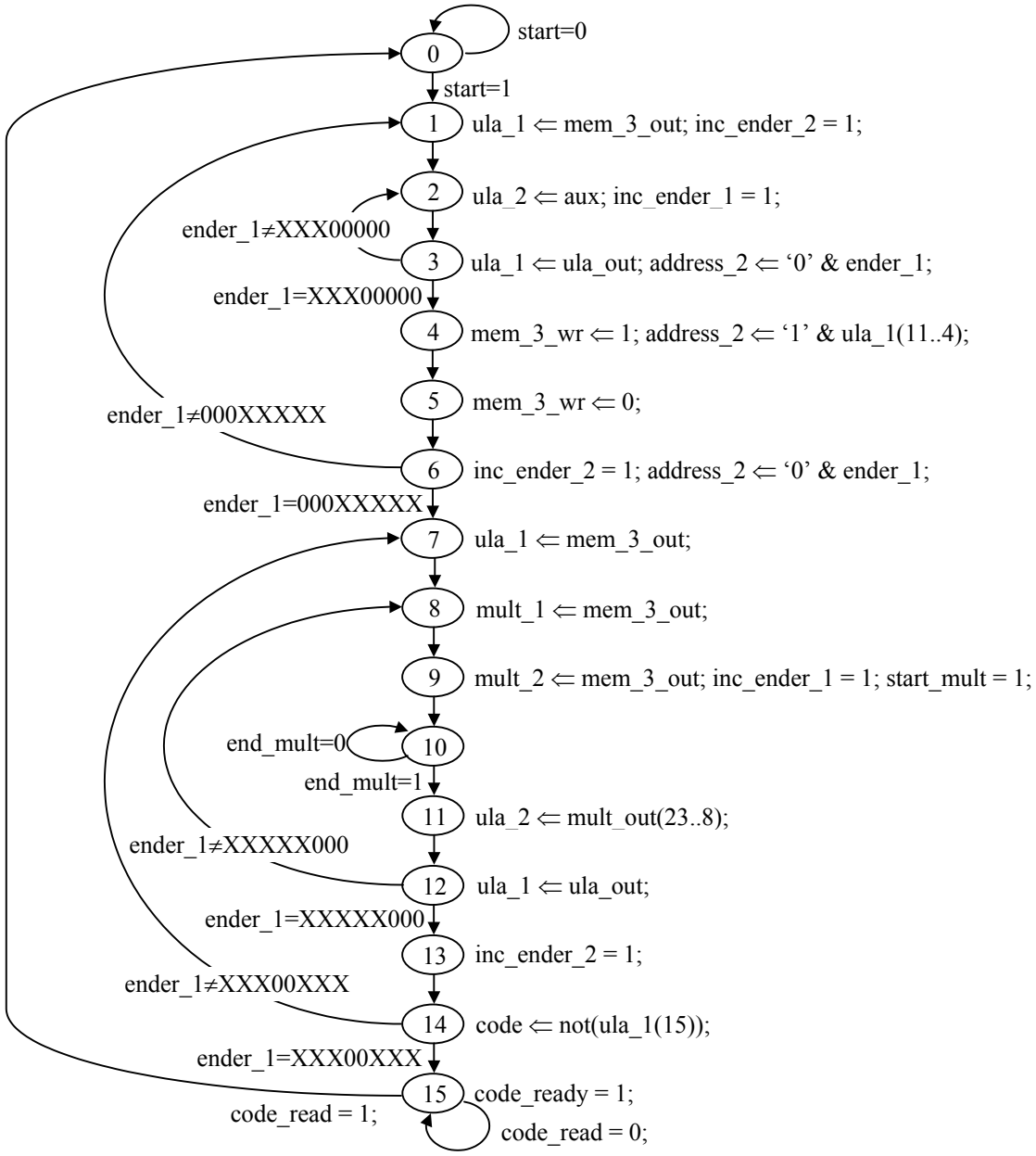


FIGURA 6.12 – Diagrama de Estados da Parte de Controle da Rede Reduzida para reconhecimento dos números

$$\text{n° ciclos (seq.)} = 2 + [(2 \cdot ne) + 4] \cdot ce + [(11 \cdot ce) + 3] \cdot cs \qquad (6.1)$$

$$\text{n° ciclos (par.)} = 2 + [(2 \cdot ne) + 4] \cdot ce + [(5 \cdot ce) + 3] \cdot cs \qquad (6.2)$$

6.3.4 Resultados de Síntese

A descrição VHDL da rede neural foi sintetizada com o software MAX+plus II [ALT92] da Altera Corporation na sua versão estudante

A tabela 6.7 apresenta os resultados de área e desempenho da síntese de todos os componentes externos da rede neural.

TABELA 6.7 – Resultado de Síntese dos Componentes Externos à Rede

Componente	Tipo	Área (Logic Cells)	Área (% FPGA)	Tempo de ciclo/atraso	Frequência (MHz)
Gerador de endereços	Contador síncrono	24	2 %	22,4 ns	44,64
Somador 16 bits	Ripple-Carry	30	2 %	68 ns	14,70
Multiplicador 12 bits	Booth seqüencial	96	8 %	92,2 ns	10,84
Componente: EPF10K20 RC240-4					

As tabelas 6.8 e 6.9 resumem os resultados de síntese da rede neural.

TABELA 6.8 - Resultado de Síntese da Rede Neural

Somador	*Multiplicador*	*Área (Logic Cells)*	*Área (% FPGA)*	*Tempo de ciclo*	*Frequência (MHz)*
Ripple-Carry	Seqüencial	303	26 %	102,0 ns	9,80
Componente: EPF10K20 RC240-4					

TABELA 6.9 - Resultado de Desempenho da Rede Neural

Somador	*Multiplicador*	*Tempo de ciclo*	*Frequência (MHz)*	*Número de ciclos*	*Tempo de classificação*
Ripple-Carry	Seqüencial	102,0 ns	9,80	910	92,82 µs
Componente: EPF10K20 RC240-4					

7 Conclusões e Perspectivas Futuras

7.1 Conclusão

O presente trabalho apresentou um conjunto de técnicas de processamento, análise e interpretação de imagem objetivando o aperfeiçoamento de um sistema de reconhecimento automático de caracteres alfanuméricos em imagens de veículos. Através do estudo de diversas técnicas existentes na literatura foram feitas experimentações que tornaram possíveis melhorar o sistema original, SIAV 1.0.

Podemos destacar as seguintes contribuições que fizeram o SIAV 2.0, uma versão aperfeiçoada.

a) A etapa de pré-processamento acrescentada ao sistema trouxe bons resultados no reconhecimento da placa e não houve um acréscimo significativo no tempo de execução.

b) O estudo estatístico realizado possibilitou a redução da área a ser pesquisada em busca da placa do automóvel, apesar de ter sido menos eficiente do que o esperado no caso dos bancos de imagens 2 e 3, que não apresentam uniformidade na sua forma de obtenção.

A etapa de reconhecimento dos caracteres foi melhorada através de estudos de técnicas de inteligência artificial e implementação de uma rede neural com topologia diferente da que aparece na primeira versão. Esta rede neural apresenta uma maior taxa de reconhecimento que a anterior, possibilitando assim uma taxa de acerto dos sete caracteres presentes na placa superior.

Os resultado apresentados pelo sistema são animadores. Em relação ao que tange o processamento de imagem propriamente dito, o objetivo foi alcançado. Em relação à arquitetura proposta para a rede neural utilizada pelo SIAV 2.0 surgiu o problema da capacidade de armazenamento do FPGA. A sugestão seria implementá-la em um dispositivo APEX 20K onde a capacidade de memória existente é suficiente tanto para a rede de letras quanto para a rede de números.

7.2 Perspectivas de Trabalhos Futuros

O trabalho apresentado constitui um estudo dentro do contexto de identificação automática de veículos através do processamento da imagem com vistas à localização e reconhecimento da placa de licença do mesmo. Como tal, pretende explorar os diferentes caminhos de implementação de um sistema desta natureza. Por este motivo, diversas modificações podem ser sugeridas como trabalho futuro.

Uma sugestão simples seria na etapa de localização da placa onde poder-se-ia gerar uma imagem negativa da imagem binarizada por software ou hardware para permitir que o sistema procure placas de fundo escuro com caracteres claros. A etapa de localização dos dígitos seria feita duas vezes, uma para cada imagem, aumentando o tempo de processamento, porém, permitiria que o sistema trabalhasse com qualquer tipo de placa.

Para o terceiro banco de imagens, capturado por pardais, seria importante uma análise das imagens para que se realizasse um pré-processamento, como por exemplo a subtração da média, visando uma melhor taxa de reconhecimento.

Na etapa de extração dos caracteres poder-se-ia extraí-los da imagem binarizada sem redimensioná-los, e descrevê-los para a rede neural através de suas formas, por exemplo, através da descrição dos caracteres através de coeficientes de Fourier tem como vantagens as características da transformada, como invariância à rotação, ao escalonamento, etc.

Outra sugestão seria o desenvolvimento de uma interface para o sistema que permitisse a utilização deste de forma interativa. O usuário, que poderiam ser funcionários de empresas responsáveis pelo reconhecimento e processamento das placas como a PROCEMPA, por exemplo, poderia optar pelos tipos de filtragens realizadas durante o pré-processamento de acordo com o tipo de imagem que deseja processar. Caso a placa possua caracteres escuros e fundo claro e vice versa, além de outros casos.

E, finalmente, a prototipação da arquitetura da rede neural de reconhecimento com o dispositivo FPGA apropriado possibilitando, assim, uma taxa de reconhecimento acelerada por hardware das placas, relativa à simulada em software.

Anexo 1 - Características dos Componentes

De acordo com os dados obtidos no Data Sheets fornecido pela Altera [MAT96] tem-se os seguintes dados.

TABELA A1 - Tabela Dados Apex20k

Table 1. APEX 20KC Device Overview (1.8V)

Device	EP20K100C	EP20K200C	EP20K400C	EP20K600C	EP20K1000C	EP20K1500C
Typical Gates	100,000	200,000	400,000	600,000	1 million	1.5 million
Maximum System Gates	269,912	525,824	1,051,648	1,537,024	1,771,520	2,391,184
Logic Elements	4,160	8,320	16,640	24,320	38,400	51,840
Maximum RAM Bits	53,248	106,496	212,992	311,296	327,680	442,368

TABELA A2 - Tabela Dados Flex10k

Table 1. FLEX 10K Device Overview (Part 1)

Feature	EPF10K10 EPF10K10A	EPF10K20	EPF10K30 EPF10K30A EPF10K30E	EPF10K40	EPF10K50 EPF10K50V EPF10K50E EPF10K50S
Typical gates (logic and RAM)	10,000	20,000	30,000	40,000	50,000
Logic elements (LEs)	576	1,152	1,728	2,304	2,880
Logic array blocks (LABs)	72	144	216	288	360
Embedded array blocks (EABs)	3	6	6	8	10
Total RAM bits	6,144	12,288	12,288 24,576	16,384	20,480 40,960

Anexo 2 - Implementação dos Blocos de Hardware em Fpga para a Rede Neural Siav 2.0

I Somador Ripple_Carry

O circuito aritmético utilizado é do tipo ripple_carry [TRI95], o mais simples entre os somadores.

A seguir são apresentados os resultados de simulação do somador.

TABELA B1 – Dados da Simulação do Somador

	Dado (bits)	*Área do FPGA (Logic Cells)*	*Área do FPGA*	*Tempo de Resposta*
Somador Ripple_Carry	24	47	8%	15 ns

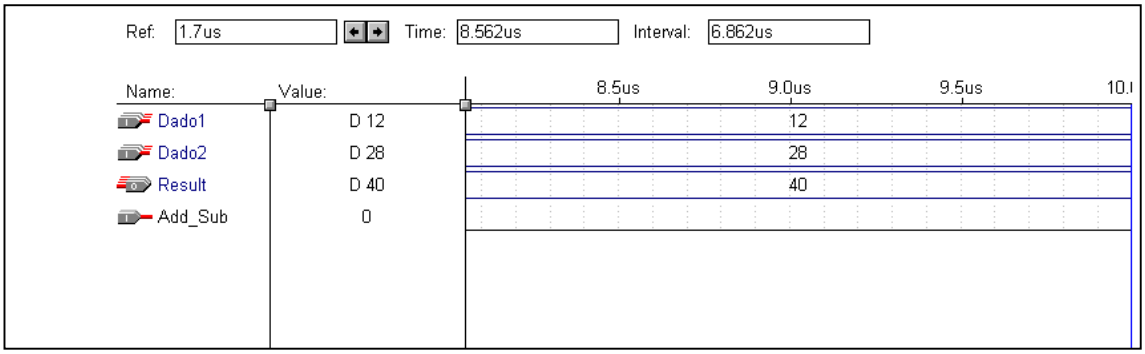


FIGURA B1 – Dados da simulação do multiplicador

II Multiplicador Booth

Existem duas maneiras de se acelerar uma multiplicação: modificar o circuito de soma de produtos parciais de maneira que o atraso seja menor, ou reduzir o número de somas necessárias.

A redução de operações de soma é conseguida através da varredura prévia dos bits do número multiplicador (MR), onde é possível verificar qual número, múltiplo do multiplicando (MD), precisa ser somado ao produto parcial.

Neste trabalho um multiplicador Booth de 12 bits foi implementado. Este multiplicador realiza uma varredura de 3 em 3 bits. O número de operações de soma se reduz a seis. A figura B2, abaixo, mostra a multiplicação através do método de varredura de 3 bits.

```
MD   0 0 1 0  0 0 0 1  1 1 0 0        (+540)
MR   0 0 0 0  1 0 0 1  0 0 1 1        (+147)

                          1 1 0     0 0 0 0 0 0 0 0 0 0 0 0
                        0 0 1       0 0 1 0 0 0 0 1 1 1 0 0
                      0 1 0       1 1 1 1 0 1 1 1 1 0 0 1 0 0
                    1 0 0         0 0 1 0 0 0 0 1 1 1 0 0
                  0 0 1         0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0
                0 0 0           0 0 1 0 0 0 0 1 1 1 0 0
                              0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0
                              1 0 1 1 1 1 0 0 1 0 0 0
                            1 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0
                            0 0 1 0 0 0 0 1 1 1 0 0
                          0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 1 0 1 0 0
```

FIGURA B2 – Multiplicação de números de 12 bits com sinal pelo algoritmo de Booth

A tabela da figura B2 apresenta as ações necessárias sobre o produto parcial de acordo com os bits do multiplicador.

TABELA B2 – Ações Sobre o Produto Parcial para o Algoritmo de Booth

MR_{i+1} , MR_i	MR_{i-1}	*Ação*
00	0	Deslocar PP
00	1	Somar MD; Deslocar PP 2 bits
01	0	Somar MD; Deslocar PP 2 bits
01	1	Somar 2*MD; Deslocar PP 2 bits
10	0	Subtrair 2*MD; Deslocar PP 2 bits
10	1	Subtrair MD; Deslocar PP 2 bits
11	0	Subtrair MD; Deslocar PP 2 bits
11	1	Deslocar PP 2 bits

Na varredura dos bits menos significativos do multiplicador deve ser considerada a existência de um zero virtual à direita do bit menos significativo.

O multiplicador Booth apresentado realiza a multiplicação de dois operandos de 12 bits. A figura B3 mostra o diagrama de estados do multiplicador implementado, com um estado de inicialização, 6 estados para o cálculo do resultado e um estado de finalização. O circuito é comandado por um sinal de **Iniciar**, que ao ir para o nível alto,

ativa o processo de multiplicação. Ao final do cálculo, o sinal **Válido** vai para o nível alto indicando que um valor correto está presente na saída **Result**.

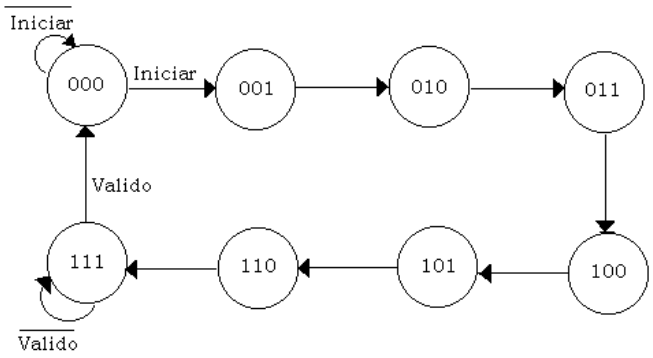


FIGURA B3 – Diagrama da Máquina de estados do multiplicador tipo Booth

A parte operativa do multiplicador está mostrada na figura B4. Consiste basicamente em um registrador de deslocamento que recebe os resultados parciais fornecidos pelo somador/subtrator.

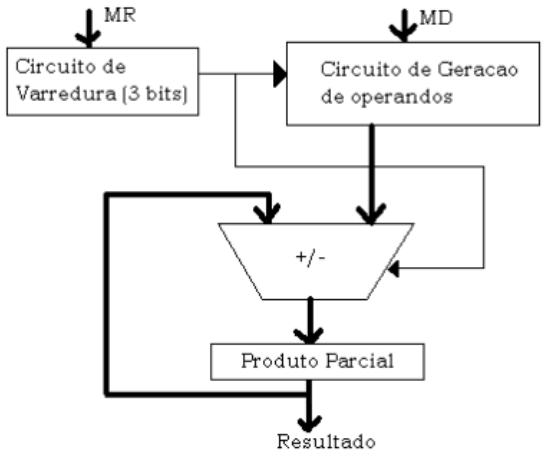


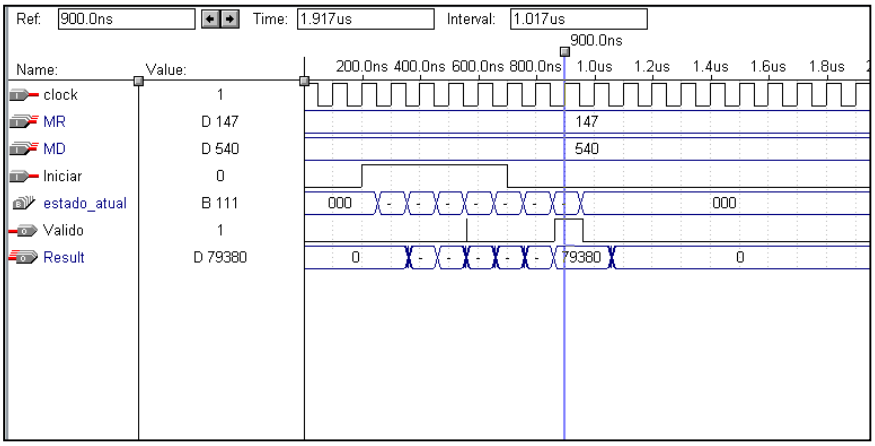
FIGURA B4 – Estrutura do Multiplicador Booth

A seguir são apresentados os resultados de simulação do multiplicador.

TABELA B3 – Dados da Simulação do Muliplicador

	Dado (bits)	*Área do FPGA (Logic Cells)*	*Área do FPGA*	*F_{max}(MHz)*	*Tempo de Resposta*
Booth	12	96	16%	10,3	679,0 ns

FIGURA B5 – Simulação do Multiplicador



III Gerador de Endereço

Como o acesso aos dados das memórias não é feito em endereços aleatórios, e sim consecutivos, a geração de endereços pode ser feita por contadores crescentes, de fácil implementação em FPGA.A descrição VHDL dos mesmos é simples e será apresentada no anexo C.

TABELA B4 – Dados da Simulação do Gerador

	Área do FPGA (Logic Cells)	*Área do FPGA*	F_{max}*(MHz)*
Gerador	34	5%	121

Anexo 3 Implementação em Vhdl dos Blocos Necessários e da Rede Neural Siav 2.0

As descrições dos elementos da rede reduzida, com multiplicador Booth seqüencial, o gerador de endereços, o somador bem como a rede são apresentadas a seguir:

Gerador de Endereços:

```
-------------------------------------------------------------------------------------
-- Disseração de Mestrado                                                          --
-- Universidade do Rio Grande do Sul - Instituto de Computação                     --
-- Circuito gerador de endereços                                                   --
-- Saídas para 2 endereços, um de 6 bits e outro de 8 bits                         --
-- Tatiane Jesus de Campos                                                         --
-------------------------------------------------------------------------------------
library ieee;
use ieee.std_logic_arith.all;
library ieee;
use ieee.std_logic_1164.all;
entity ender is
 port (
       clock  : in bit;                    -- Sinal de sincronismo
       rst    : in bit;                   -- Sinal de inicialização
       hab1   : in bit;                    -- Sinal de incremento do endereço 1
       ender1 : out std_logic_vector(7 downto 0); -- Endereço 1a
       hab2   : in bit;                    -- Sinal de incremento do endereço 2
       ender2 : out std_logic_vector(5 downto 0) -- Endereço 2a
     );
end ender;
architecture Sender of ender is
  signal end1: integer range 0 to 15;
  signal end2: integer range 0 to 15;
begin
-- Processo de controle do endereço 1
  process(clock, hab1)
    begin
      if (clock = '0' and not clock'stable) then
        if rst = '0' then
          if hab1 = '1' then
            end1 <= end1 + 2;
          else
            end1 <= end1;
          end if;
        else
          end1 <= 0;
        end if;
      end if;
  end process;
-- Processo de controle do endereço 2
  process(clock, hab2)
    begin
      if (clock = '0' and not clock'stable) then
        if rst = '0' then
          if hab2 = '1' then
            end2 <= end2 + 2;
          else
            end2 <= end2;
          end if;
        else
```

```
        end2 <= 0;
       end if;
     end if;
  end process;
-- Conversão de dados do tipo inteiro para std_logic_vector
  ender1 <= conv_std_logic_vector(end1,8);
  ender2 <= conv_std_logic_vector(end2,6);
end Sender;
```

Multiplicador Booth de 12 x 12 bits:

```
-------------------------------------------------------------------------------------
-- Disseração de Mestrado                                                          --
-- Universidade do Rio Grande do Sul - Instituto de Computação                     --
-- Multiplicador Booth para 12 x 12 bits Usando Somador Riple Carry                --
-- Tatiane Jesus de Campos                                                         --
-------------------------------------------------------------------------------------
entity booth12x12 is
  port(
       Clock   : in bit;                    -- Sinal de sincronismo
       Iniciar : in bit;                    -- Controle para o início da multiplicação
       MR      : in bit_vector(11 downto 0);   -- Primeiro operando
       MD      : in bit_vector(11 downto 0);  -- Segundo operando
       Result  : out bit_vector(24 downto 0); -- Resultado da multiplicação
       Valido  : out bit                    -- Indicação de término do cálculo
     );
end booth12x12;

architecture Sbooth of booth12x12 is
  type estados is (estado0, estado1, estado2, estado3, estado4, estado5, estado6, estado7);
  signal estado_atual   : estados;
  signal proximo_estado : estados;
  signal scan3          : bit_vector(2 downto 0);
  signal alu2           : bit_vector(12 downto 0);
  signal alu1           : bit_vector(12 downto 0);
  signal out_alu        : bit_vector(12 downto 0);
  signal carry          : bit_vector(11 downto 0);
  signal shiftpp        : bit;
  signal passamd        : bit;
  signal shiftmd        : bit;
  signal PP             : bit_vector(24 downto 0);
  signal in22           : bit_vector(14 downto 0);  --> Atenção
  signal zero           : bit;


begin
------------------------------- Máquina com 8 estados ----------------------------------
-- Sempre que a entrada Iniciar é 1 no estado0 a máquina avança para os próximos estados
  process(estado_atual, Iniciar)
begin
     case estado_atual is
       when estado0 =>
         if (Iniciar='0') then
           proximo_estado <= estado0;
         else
           proximo_estado <= estado1;
         end if;
       when estado1 =>
         proximo_estado <= estado2;
       when estado2 =>
         proximo_estado <= estado3;
```

```
        when estado3 =>
          proximo_estado <= estado4;
                    when estado4 =>
          proximo_estado <= estado5;
                    when estado5 =>
          proximo_estado <= estado6;
                    when estado6 =>
          proximo_estado <= estado7;
        when estado7 =>
          proximo_estado <= estado0;
      end case;
  end process;

-- Mudança de estado na borda de descida do relógio
  process(Clock)
    begin
      if (Clock = '0' and not Clock'stable) then
        estado_atual <= proximo_estado;
      end if;
  end process;

-- O sinal de resultado válido ocorre no último estado (estado5)
  process(Clock)
    begin
      if (estado_atual = estado7) then
        Valido <= '1';
      else
        Valido <= '0';
      end if;
  end process;

------------------------------ Final da máquina de estados -------------------------------
------------------------------ Início da parte operativa ---------------------------------
  zero <= '0';
-- Mux para o scanning dos bits do multiplicador (MR)
  process (clock)
    begin
      case estado_atual is
        when estado1 =>
          scan3 <= MR(1) & MR(0) & '0';
        when estado2 =>
          scan3 <= MR(3) & MR(2) & MR(1);
                    when estado3 =>
          scan3 <= MR(5) & MR(4) & MR(3);
        when estado4 =>
          scan3 <= MR(7) & MR(6) & MR(5);
                    when estado5 =>
          scan3 <= MR(9) & MR(8) & MR(7);
        when estado6 =>
          scan3 <= MR(11) & MR(10) & MR(9);
                    when others =>
          scan3 <= B"000";
      end case;
  end process;

-- Sinal de deslocamento para o registrador de produtos parciais
  process(Clock)
    begin
      case estado_atual is
        when estado0 | estado7 =>
```

```
        shiftpp <= '0';
      when others =>
        shiftpp <= '1';
    end case;
  end process;
-- Circuito para a seleção do operando 2 do somador
  passamd <= scan3(1) xor scan3(0);
  shiftmd <= (scan3(2) nor (scan3(1) nand scan3(0))) or (scan3(2) and (scan3(1) nor scan3(0)));
  alu2(0)  <= scan3(2) xor ((MD(0)  and passamd) or zero);
  alu2(1)  <= scan3(2) xor ((MD(1)  and passamd) or (MD(0)  and shiftmd));
  alu2(2)  <= scan3(2) xor ((MD(2)  and passamd) or (MD(1)  and shiftmd));
  alu2(3)  <= scan3(2) xor ((MD(3)  and passamd) or (MD(2)  and shiftmd));
  alu2(4)  <= scan3(2) xor ((MD(4)  and passamd) or (MD(3)  and shiftmd));
  alu2(5)  <= scan3(2) xor ((MD(5)  and passamd) or (MD(4)  and shiftmd));
  alu2(6)  <= scan3(2) xor ((MD(6)  and passamd) or (MD(5)  and shiftmd));
  alu2(7)  <= scan3(2) xor ((MD(7)  and passamd) or (MD(6)  and shiftmd));
  alu2(8)  <= scan3(2) xor ((MD(8)  and passamd) or (MD(7)  and shiftmd));
  alu2(9)  <= scan3(2) xor ((MD(9)  and passamd) or (MD(8)  and shiftmd));
  alu2(10) <= scan3(2) xor ((MD(10) and passamd) or (MD(9)  and shiftmd));
  alu2(11) <= scan3(2) xor ((MD(11) and passamd) or (MD(10) and shiftmd));
  alu2(12) <= scan3(2) xor ((MD(11) and passamd) or (MD(11) and shiftmd));

-- Operando 1 do somador
  alu1 <= PP(24 downto 12);
-- Somador tipo ripple carry (Full adders)
  out_alu(0) <= (alu1(0) xor alu2(0)) xor scan3(2);
  carry(0)   <= (alu1(0) and alu2(0)) or (scan3(2) and (alu1(0) xor alu2(0)));
  out_alu(1) <= (alu1(1) xor alu2(1)) xor carry(0);
  carry(1)   <= (alu1(1) and alu2(1)) or (carry(0) and (alu1(1) xor alu2(1)));
  out_alu(2) <= (alu1(2) xor alu2(2)) xor carry(1);
  carry(2)   <= (alu1(2) and alu2(2)) or (carry(1) and (alu1(2) xor alu2(2)));
  out_alu(3) <= (alu1(3) xor alu2(3)) xor carry(2);
  carry(3)   <= (alu1(3) and alu2(3)) or (carry(2) and (alu1(3) xor alu2(3)));
  out_alu(4) <= (alu1(4) xor alu2(4)) xor carry(3);
  carry(4)   <= (alu1(4) and alu2(4)) or (carry(3) and (alu1(4) xor alu2(4)));
  out_alu(5) <= (alu1(5) xor alu2(5)) xor carry(4);
  carry(5)   <= (alu1(5) and alu2(5)) or (carry(4) and (alu1(5) xor alu2(5)));
  out_alu(6) <= (alu1(6) xor alu2(6)) xor carry(5);
  carry(6)   <= (alu1(6) and alu2(6)) or (carry(5) and (alu1(6) xor alu2(6)));
  out_alu(7) <= (alu1(7) xor alu2(7)) xor carry(6);
  carry(7)   <= (alu1(7) and alu2(7)) or (carry(6) and (alu1(7) xor alu2(7)));
  out_alu(8) <= (alu1(8) xor alu2(8)) xor carry(7);
  carry(8)   <= (alu1(8) and alu2(8)) or (carry(7) and (alu1(8) xor alu2(8)));
  out_alu(9) <= (alu1(9) xor alu2(9)) xor carry(8);
  carry(9)   <= (alu1(9) and alu2(9)) or (carry(8) and (alu1(9) xor alu2(9)));
 out_alu(10)<= (alu1(10) xor alu2(10)) xor carry(9);
  carry(10)  <= (alu1(10) and alu2(10)) or (carry(9) and (alu1(10) xor alu2(10)));
  out_alu(11)<= (alu1(11) xor alu2(11)) xor carry(10);
  carry(11)  <= (alu1(11) and alu2(11)) or (carry(10) and (alu1(11) xor alu2(11)));
  out_alu(12)<= (alu1(12) xor alu2(12)) xor carry(11);
  in22 <= out_alu(12) & out_alu(12) & out_alu;

-- Registrador para armazenamento dos produtos parciais e resultado final
  process (Clock)
    begin
      if Clock='0' and not Clock'stable then
        if shiftpp = '1' then
          for i in 0 to 9 loop
            PP(i) <= PP(i+2);
          end loop;
```

```
                    PP(24 downto 10) <= in22;
      else
        if estado_atual=estado0 then
          PP <= "000000000000000000000000";
        end if;
      end if;
    end if;
  end process;

  Result <= PP(24 downto 0);
end Sbooth;
```

Somador / Subtrator de 16 bits Ripple_Cary:

```
---------------------------------------------------------------------------------------------
-- Dissertação de Mestrado
-- Universidade do Rio Grande do Sul - Instituto de Computação
-- Circuito somador/subtrator tipo ripple carry
-- Operandos de 16 bits
--  Tatiane Jesus de Campos
---------------------------------------------------------------------------------------------
entity somador16 is
  port(
       Dado1   : in bit_vector(15 downto 0);  -- Primeiro operando
       Dado2   : in bit_vector(15 downto 0);  -- Segundo operando
       Add_Sub : in  bit;                     -- Soma:'0'; Subtração:'1'
       Result  : out bit_vector(15 downto 0)  -- Resultado da operação
     );
end somador16;

architecture Saddsub16 of somador16 is
  signal aux2   : bit_vector(15 downto 0);
  signal vai_um : bit_vector(14 downto 0);

begin
-- Inversor controlado
  aux2(0)  <= Dado2(0)  xor Add_Sub;
  aux2(1)  <= Dado2(1)  xor Add_Sub;
  aux2(2)  <= Dado2(2)  xor Add_Sub;
  aux2(3)  <= Dado2(3)  xor Add_Sub;
  aux2(4)  <= Dado2(4)  xor Add_Sub;
  aux2(5)  <= Dado2(5)  xor Add_Sub;
  aux2(6)  <= Dado2(6)  xor Add_Sub;
  aux2(7)  <= Dado2(7)  xor Add_Sub;
  aux2(8)  <= Dado2(8)  xor Add_Sub;
  aux2(9)  <= Dado2(9)  xor Add_Sub;
  aux2(10) <= Dado2(10) xor Add_Sub;
  aux2(11) <= Dado2(11) xor Add_Sub;
  aux2(12) <= Dado2(12) xor Add_Sub;
  aux2(13) <= Dado2(13) xor Add_Sub;
  aux2(14) <= Dado2(14) xor Add_Sub;
  aux2(15) <= Dado2(15) xor Add_Sub;
-- Somadores completos (Full adders)
  Result(0)  <= (Dado1(0) xor aux2(0)) xor Add_Sub;
  vai_um(0)  <= (Dado1(0) and aux2(0)) or (Add_Sub and (Dado1(0) xor aux2(0)));
  Result(1)  <= (Dado1(1) xor aux2(1)) xor vai_um(0);
  vai_um(1)  <= (Dado1(1) and aux2(1)) or (vai_um(0) and (Dado1(1) xor aux2(1)));
  Result(2)  <= (Dado1(2) xor aux2(2)) xor vai_um(1);
  vai_um(2)  <= (Dado1(2) and aux2(2)) or (vai_um(1) and (Dado1(2) xor aux2(2)));
  Result(3)  <= (Dado1(3) xor aux2(3)) xor vai_um(2);
  vai_um(3)  <= (Dado1(3) and aux2(3)) or (vai_um(2) and (Dado1(3) xor aux2(3)));
```

```
  Result(4)  <= (Dado1(4) xor aux2(4)) xor vai_um(3);
  vai_um(4)  <= (Dado1(4) and aux2(4)) or (vai_um(3) and (Dado1(4) xor aux2(4)));
  Result(5)  <= (Dado1(5) xor aux2(5)) xor vai_um(4);
  vai_um(5)  <= (Dado1(5) and aux2(5)) or (vai_um(4) and (Dado1(5) xor aux2(5)));
  Result(6)  <= (Dado1(6) xor aux2(6)) xor vai_um(5);
  vai_um(6)  <= (Dado1(6) and aux2(6)) or (vai_um(5) and (Dado1(6) xor aux2(6)));
  Result(7)  <= (Dado1(7) xor aux2(7)) xor vai_um(6);
  vai_um(7)  <= (Dado1(7) and aux2(7)) or (vai_um(6) and (Dado1(7) xor aux2(7)));
  Result(8)  <= (Dado1(8) xor aux2(8)) xor vai_um(7);
  vai_um(8)  <= (Dado1(8) and aux2(8)) or (vai_um(7) and (Dado1(8) xor aux2(8)));
  Result(9)  <= (Dado1(9) xor aux2(9)) xor vai_um(8);
  vai_um(9)  <= (Dado1(9) and aux2(9)) or (vai_um(8) and (Dado1(9) xor aux2(9)));
  Result(10) <= (Dado1(10) xor aux2(10)) xor vai_um(9);
  vai_um(10) <= (Dado1(10) and aux2(10)) or (vai_um(9) and (Dado1(10) xor aux2(10)));
  Result(11) <= (Dado1(11) xor aux2(11)) xor vai_um(10);
  vai_um(11) <= (Dado1(11) and aux2(11)) or (vai_um(10) and (Dado1(11) xor aux2(11)));
  Result(12) <= (Dado1(12) xor aux2(12)) xor vai_um(11);
  vai_um(12) <= (Dado1(12) and aux2(12)) or (vai_um(11) and (Dado1(12) xor aux2(12)));
  Result(13) <= (Dado1(13) xor aux2(13)) xor vai_um(12);
  vai_um(13) <= (Dado1(13) and aux2(13)) or (vai_um(12) and (Dado1(13) xor aux2(13)));
  Result(14) <= (Dado1(14) xor aux2(14)) xor vai_um(13);
  vai_um(14) <= (Dado1(14) and aux2(14)) or (vai_um(13) and (Dado1(14) xor aux2(14)));
  Result(15)  <= (Dado1(15) xor aux2(15)) xor vai_um(14);

end Saddsub16;
```

Rede Neural (caracter de entrada 32 bits (8x4)):

```
--------------------------------------------------------------------------------------------
-- Dissertação de Mestrado
-- Universidade do Rio Grande do Sul - Instituto de Computação
-- Rede Neural para reconhecimento de numeros:
 --  Tatiane Jesus de Campos
-- Circuito Rede neural com 32 entradas, 8 neuronios escondidos e 10 saidas
--------------------------------------------------------------------------------------------
library lpm;
use lpm.lpm_components.all;
entity rede8x4 is
  port(
       clock     : in bit;                    -- Sinal de sincronismo
       reset     : in bit;                    -- Sinal de inicialização
       x_in      : in bit_vector(2 downto 0);   -- Entrada para os dados amostrados
       w_e       : in bit;                    -- Sinal de escrita na memória de dados
       treina    : in bit;                    -- Convolução:'0'; Adaptação:'1'
       y         : out bit_vector(9 downto 0);  -- Saida do filtro
       y_valido  : out bit;                     -- Indicação de término do cálculo
       e         : out bit_vector(9 downto 0)   -- Saída do valor de erro na adaptação
     );
end rede8x4;

architecture SFiltro of rede8x4 is
------------------------------ Componentes externos ao filtro -----------------------------
-- Elementos de memória do Filtro
component lpm_ram_dq
   generic (
            LPM_WIDTH          : POSITIVE;
            LPM_WIDTHAD        : POSITIVE;
            LPM_FILE           : STRING;
            LPM_INDATA         : STRING;
            LPM_ADDRESS_CONTROL : STRING;
```

```
          LPM_OUTDATA         : STRING
         );
  port (
        data     : in bit_vector(LPM_WIDTH-1 downto 0);
        address  : in bit_vector(LPM_WIDTHAD-1 downto 0);
        we       : in bit;
        inclock  : in bit;
        outclock : in bit;
        q        : out bit_vector(LPM_WIDTH-1 downto 0)
       );
end component;

-- Gerador de endereços
component ender
  port (
            clock  : in bit;
        rst    : in bit;
                        hab1   : in bit;
        ender1 : out bit_vector(7 downto 0);
        hab2   : in bit;
        ender2 : out bit_vector(5 downto 0);
      );
end component;

-- Multiplicador Booth12x12
component booth12x12
  port(
       clock   : in bit;
       Iniciar : in bit;
       MR      : in bit_vector(11 downto 0);
       MD      : in bit_vector(11 downto 0);
       Result  : out bit_vector(24 downto 0);
       Valido  : out bit
     );
end component;

-- Somador/Subtrator Riple Carry de 16 bits
component somador16
  port(
       Dado1   : in bit_vector(15 downto 0);
       Dado2   : in bit_vector(15 downto 0);
       Add_Sub : in  bit;
       Result  : out bit_vector(15 downto 0)
     );
end component;
---------------- Declaração dos sinais internos e registradores ----------------------
-- Sinais da máquina de estados
  type estados is (est0, est1, est2, est3, est4, est6, est7,
                   est8, est9, est10, est11, est12, est13, est14, est15);
  signal estado_atual   : estados;
  signal proximo_estado : estados;
  signal y_val          : bit;
  signal ender_zero     : bit;
-- Sinais do gerador de endereços
  signal inc_ender_1    : bit;
  signal inc_ender_2    : bit;

-- Sinais das memórias
  signal ender_1        : bit_vector(7 downto 0);
  signal coef_out       : bit_vector(19 downto 0);
```

```
  signal wr_ender_1     : bit;
  signal ender_2        : bit_vector(5 downto 0);
  signal ula_1          : bit_vector(2 downto 0);
  signal matriz_add     : bit_vector(2 downto 0);
-- Sinais do multiplicador
  signal inicio_mult    : bit;
  signal mult_1         : bit_vector(11 downto 0);
  signal mult_2         : bit_vector(11 downto 0);
  signal mult_out       : bit_vector(24 downto 0);
  signal fim_mult       : bit;
-- Sinais do Somador/Subtrator
  signal ula_1          : bit_vector(15 downto 0);
  signal ula_2          : bit_vector(15 downto 0);
  signal ula_out        : bit_vector(15 downto 0);
  signal ula_oper       : bit;
-- Registrador da saída Y, do erro e do coeficiente corrigido
  signal Y_out          : bit_vector(24 downto 0);
  signal erro           : bit_vector(19 downto 0);
  signal coef_in        : bit_vector(19 downto 0);
begin
---------------------------------- Máquina de estados ----------------------------------
-- Sempre que a entrada Reset é 0 no estado0 a máquina avança para os próximos estados
  process(estado_atual, reset)
    begin
      case estado_atual is
        when est0 =>
          if (reset = '0') then
            proximo_estado <= est0;
          else
            proximo_estado <= est1;
          end if;
        when est1 =>
          proximo_estado <= est2;
        when est2 =>
          proximo_estado <= est3;
        when est3 =>
          proximo_estado <= est4;
        when est4 =>
          if (ender_1 != 'XFF') then
            proximo_estado <= est3;
          else
            proximo_estado <= est5;
          end if;
        when est5 =>
            proximo_estado <= est6;
       when est6 =>
            proximo_estado <= est7;
        when est7 =>
            proximo_estado <= est8;
        when est8 =>
            if (ender_1 != 'AFF') then
              proximo_estado <= est2;
           else
              proximo_estado <= est9;
          end if;
        when est9 =>
          proximo_estado <= est10;
        when est10 =>
                        if (valido = '0') then
            proximo_estado <= est10;
```

```
        else
          proximo_estado <= est11;
        end if;
      when est11 =>
            proximo_estado <= est12;
      when est12 =>
        if (ender_2 != '0x14') then
          proximo_estado <= est10;
        else
          proximo_estado <= est13;
        end if;
      when est13 =>
        proximo_estado <= est14;
      when est14 =>
        proximo_estado <= est15;
      when est15 =>
        proximo_estado <= est16;
      when est16 =>
        if (ender_3 != '0x1E') then
          proximo_estado <= est9;
        else
          proximo_estado <= est17;
        end if;
      when est17 =>
        if (code_ready = '0') then
          proximo_estado <= est17;
        else
          proximo_estado <= est0;
        end if;
    end case;
  end process;

-- Mudança de estado na borda de descida do relógio
  process(clock)
    begin
      if (clock = '0' and not clock'stable) then
        estado_atual <= proximo_estado;
      end if;
  end process;
---------------------------------- Parte operativa -------------------------------------
-- Sinais de controle do gerador de endereços
  with estado_atual select
    inc_ender_1 <= '1' when est2, '1' when est11, '0' when others;
  with estado_atual select
    inc_ender_2 <= not(ender_1(4) and ender_1(3) and ender_1(2) and ender_1(1) and ender_1(0)) or
treina when est2,
                 not(ender_1(4) and ender_1(3) and ender_1(2) and ender_1(1) and ender_1(0)) when est8,
                 '0' when others;
  with estado_atual select
    inc_ender_3 <= '1' when est13, '0' when others;
-- Sinal de escrita na memória de coeficientes
  with estado_atual select
    wr_ender_1 <= '1' when est11, '0' when others;
-- Sinal de controle do multiplicador
  with estado_atual select
    inicio_mult <= '1' when est2, '1' when est8, '0' when others;

-- Sinal de controle do Somador/Subtrator
  with estado_atual select
    ula_oper <= '1' when est6, '1' when est7, '0' when others;
```

```
-- Sinal que indica o final do cálculo da saída Y
  with estado_atual select
    y_val <= '1' when est6, '0' when others;
-- Sinal que indica o endereço zero da memória de coeficientes
  ender_zero <= ender_1(0) or ender_1(1) or ender_1(2) or ender_1(3) or ender_1(4);
------------------------ Mapeamento dos elemento externos -----------------------------
-- Gerador de endereços
Enderecos: gerador
  port map (
            clock,
            reset,
            inc_ender_y,
            ender_y,
            inc_ender_1,
            ender_1,
            inc_ender_2,
            ender_2,
           );
-- Memória de coeficientes
coeficientes: lpm_ram_dq
  generic map ( 20, 5, "coef.hex", "REGISTERED", "REGISTERED", "REGISTERED")
  port map (
            coef_in,
            ender_1,
            wr_ender_1,
           clock,
            coef_out
           );
-- Memória dos dados de entrada
entrada_x: lpm_ram_dq
  generic map ( 3, 5, "amost.hex", "REGISTERED", "REGISTERED", "REGISTERED")
  port map (
            x_in,
            ender_2,
            w_e,
             clock,
            x_out
           );
-- Memória dos resultados esperados
saida_d: lpm_ram_dq
  generic map ( 3, 5, "esper.hex", "REGISTERED", "REGISTERED", "REGISTERED")
  port map (
            x_in,
            ender_3,
            w_e,
            clock,
            d_out
           );
-- Multiplicador
multiplicador: booth12x12
  port map (
            clock,
            inicio_mult,
            mult_1,
            mult_2,
            mult_out,
            fim_mult
           );
-- Somador 16
ula: somador16
```

```
  port map (
            ula_1,
            ula_2,
            ula_oper,
            ula_out
    );
---------------------- Controle dos registradores e Multiplexadores ----------------------
-- Armazenamento da entrada 2 do multiplicador
  process(clock)
    begin
      if (clock = '0' and not clock'stable) then
        case estado_atual is
          when est2 =>
            mult_2 <= coef_out;
          when est8 =>
            mult_2 <= erro;
          when others  =>
            mult_2 <= mult_2;
        end case;
      end if;
  end process;
-- Armazenamento da entrada 1 do multiplicador
  process(clock)
    begin
      if (clock = '0' and not clock'stable) then
        case estado_atual is
          when est2 | est8 =>
            mult_1 <= x_out;
          when others  =>
            mult_1 <= mult_1;
        end case;
      end if;
  end process;
-- Armazenamento da entrada 1 da ula
  process(clock)
    begin
      if (clock='0' and not clock'stable) then
        case estado_atual is
          when est3 =>
            ula_1 <= mult_out;
          when est6 =>
            ula_1 <= d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) &
d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out(2) &
d_out(2) & d_out(2) & d_out(2) & d_out(2) & d_out;
          when est9 =>
            ula_1 <= mult_out(22) & mult_out (22) & mult_out(22 downto 2); -- 0*25 erro
          when others  =>
            ula_1 <= ula_1;
        end case;
      end if;
  end process;
-- Armazenamento da entrada 2 da ula
  process(clock)
    begin
      if (clock='0' and not clock'stable) then
        case estado_atual is
          when est3 | est6 =>
            ula_2 <= Y_out;
          when est9 =>
            ula_2 <= coef_out(19) & coef_out(19) & coef_out(19) & coef_out;
```

```
          when others  =>
            ula_2 <= ula_2;
        end case;
      end if;
  end process;
-- Armazenamento da saída Y
  process(clock)
    begin
      if (clock='0' and not clock'stable) then
        case estado_atual is
          when est1 =>
            Y_out <= "00000000000000000000000";
          when est4 =>
            Y_out <= ula_out;
          when others =>
            Y_out <= Y_out;
        end case;
      end if;
  end process;
-- Armazenamento do erro
  process(clock)
    begin
      if (clock='0' and not clock'stable) then
        case estado_atual is
          when est7 =>
            erro <= ula_out(22 downto 3);
          when others =>
            erro <= erro;
        end case;
      end if;
  end process;
-- Armazenamento do coeficiente corrigido
  process(clock)
    begin
      if (clock='0' and not clock'stable) then
        case estado_atual is
          when est10 =>
            coef_in <= ula_out(22 downto 3);
          when others =>
            coef_in <= coef_in;
        end case;
      end if;
  end process;
y       <= Y_out;
y_valido <= y_val;
e       <= erro;
```

End Srede8x4;

Bibliografia

[ALT2000] ALTERA. **Data Sheet**. Disponível em: <http://www.altera.com>. Acesso em: 08 mar. 2000.

[AWC96] AWCOOCK,G. J.; THOMAS,R. **Applied Image Processing**. New York: McGraw Hill, 1996.

[BAR 99] BARROSO, J. et al. **Number plate reading using computer vision**. Disponível em: <http://www.utad.pt/~jbarroso/html/isie97.html>. Acesso em: 7 jul. 1999.

[BRA97] BRAGA, A. P.; CARVALHO, A. P. L.; LUDERMIR T. B. Fundamentos de Redes Neurais Artificiais. In: ESCOLA DE COMPUTAÇÃO 11., 1997. **Anais...** Rio de Janeiro: [s.n.], 1997.

[CAR88] CARVALHO, L. A. V. **Redes Neuronais e a Tradição Conexionista da Inteligencia Artificial**, 1988. Dissertação (Mestrado) - COPPE, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

[CAS96] CASTLEMAN, K. R. **Digital Image Processing.** [S.l.]: Prentice Hall, 1996.

[COE98] COETZEE C.; BOTHA C.; WEBER D. PC Based Number Plate Recognition System. In: INTERNATIONAL CONFERENCE ON INDUSTRIAL ELECTRONICS, 1998. **Proceedings...** [S.l. : s.n.],1998.

[COT85] COTTREL, G. W. **A Connectionist Aproach to Word Sense Disambiguation**, 1985. Tese (Doutorado), University of Rochester, Rochester.

[EPP98] EPPLER, W.; FISCHER, T.; GEMMEKE, H.; MENCHIKOV, A. High Speed Neural Network Chip for Trigger Purposes in High Energy Physics. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE, 1998, Paris. **Proceedings**... Los Alamitos, CA, USA: IEEE Computer Society, 1998. p. 108-115.

[FAC93] FACON, J. Processamento e Análise de Imagens. In: EBAI, 1993. **Anais**... Cordoba: [s.n.], 1993. 198p.

[FRA99] FRANCO, D.T. **Um Estudo Comparativo de Filtros Adaptativos em FPGA**. 1999. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[GON 93] GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. [S.l.]: Addison-Wesley, 1993.

[HAY94] HAYKIN, S. **Neural Networks:** a comprehensive foundation. New York: Macmillan College Publishing Company, 1994.

[HEB49] HEBB, D. O. **The organization of Behavior.** New York: John Wiley & Sons, 1949.

[HIG99] HIGH TECH SOLUTION. **SeeCar.** Disponível em: <http://www.htsol.com>. Acesso em: 12 set.1999.

[HOP82] HOPFIELD, J. J. **Neural Networks and Physical Systems with Emergent Computational Abilities**. Washington: [s.n.], 1982. (Procceding of the National Academy of Sciences, v.79).

[INI 85] INIGO, R. M. Traffic Monitoring and Control using Machine Vision: A Survey. **IEEE Transactions on Industrial Electronics**, New York, v. IE, p 177-185, Aug. 1985.

[JAI89] JAIN, A. K. **Fundamentos of Digital Image Processing**. [S.l.]: Prentice Hall, 1989.

[KOS92] KOSKO, B. **Neural Network for Signal Processing**. New York, Prentice Hall, 1992.

[KOV96] KOVÁCS Z. L. **Redes Neurais Artificiais:** Fundamentos e Aplicações. São Paulo: Ed. Acadêmica, 1996. 165p.

[LIB97] LIBERMAN, F. **Classificação de Imagens Digitais por Textura usando Redes Neurais.** 1997. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MAT96] MATLAB. **Using Matlab**.Version 5. Natick, MA, 1996.

[McC43] McCULLOCH, W. S.; PITTS, W. A logical Calculus of the Ideas Imminent in Nervous Activity. **Bulletin of Mathematical Biophisics**, [S.l.], p. 115 - 133, 1943.

[McC86] McCLELLAND, J.; RUMERLHART, D. **Parallel Distributed Processing**. Cambridge, MA: MIT Press, 1986.

[MIN69] MINSKY, M.; PAPERT, S. **Perceptons:** An Introduction to Computational Geometric. Cambridge: MIT Press, 1969.

[MOL98] MOLZ, R. F. **Proposta de Implementação em Hardware dedicado de Redes Neurais Competitivas com Técnicas de Circuitos Integrados Analógicos**, 1998. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[NIB86] NIBLACK, W. **An Introduction to Digital Image Processing.** New York: Prentice Hal, 1986.

[PAR85] PARKER, D. B. **Learning Logic.** Cambridge: MIT, 1985.

[PAZ88] PAZ, E. P.; CUNHA, T. N. **Iniciação ao Processamento Digital de Imagens**. Rio de Janeiro: UFRJ, 1988.

[ROS84] ROSENFELD, A. **Digital Image Processing Techniques.** Orlando, Florida: Academic Press, 1984. v.2, p. 257 - 287.

[ROS59] ROSENBLAT, R. **Principles of Neurodynamics**, New York: Spartan Books, 1959.

[RUM86] RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning Internal Representations by Error Propagation**. La Jolla: University of California at San Diego, Institute of Cognitive Science, 1986.

[SHU51] SHULDINER P. W.; D'AGOSTINO S. A.; WOODSON J. B. Determinig Detailed Origin-Destination and Travel Time Patterns Using Video and Machine Vision License Plate Matching. In: TRANSPORTATION RESEARCH RECORD, **Proceedings...**, pp. 8-17, 1951.

[SIM90] SIMPSON, P. **Artificial Neural Systems**: Foundations, Paradigms, Aplications and Implementations. New York : Pergamon Press, 1990.

[SOU 2000] SOUZA, F. **Localização e Leitura Automática de Caracteres Alfanuméricos: uma aplicação na identificação de veículos.** 2000. Dissertação (Mestrado) – IEE, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[TRI95] TRIER, O. D.; JAIN, A. K. Goal-Directed Evaluation of Binarization Methods, **IEEE Transactions on Pattern Analysys and Machine Intelligence**, New York, v. 17, n.12, p. 1191-1201, Dec. 1995.

[TUR51] TURNER S. M. Advanced Techniques for Travel Time Data Collection. In: TRANSPORTATION RESEARCH RECORD. **Proceedings...**, p. 51-58, 1951.

[WID62] WIDROW, B. Generalization and Information Storage in Network of ADALINE Neurons, In: SELF-ORGANIZATION SYSTEM. WASHINGTON, **Procceding...** Spartan Books, 1962.